NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DESIGN OF AN INTELLIGENT
TUTORING SYSTEM SHELL

by

Robert E. Scurlock Jr.

September, 1991

Thesis Advisor: Yuh-jeng Lee

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Computer Science Dept. Naval Postgraduate School; 6b. OFFICE SYMBOL (if applicable): CS	7a. NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
DESIGN OF AN INTELLIGENT TUTORING SYSTEM SHELL(U)

12. PERSONAL AUTHOR(S)
Scurlock, Robert E. Jr.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM 08/89 TO 09/91	September 1991	79

16. SUPPLEMENTARY NOTATION: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Intelligent Tutoring System, Intelligent Training System, Intelligent Computer Aided Instruction, Intelligent Tutoring System Shell

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Computer technology has brought about numerous changes in the availability of educational media, especially the Intelligent Tutoring System (ITS). Since the development of an ITS is such an interdisciplinary task, the instructor needs assistance in developing these educational aides. An ITS shell, or authoring system, is the tool that will enable ITSs to make the transition from research arena and into the educational environment.

The conceptual model of the ITS shell proposed in this thesis uses a layered approach to accessing the different modules of the ITS. The components, or subcomponents, of each module consist of either existing programs, or are selectable options developed by area experts. These options should allow the instructor to develop an ITS concentrating on the material being presented and on the method of interaction the student has with that material. The emphasis on the construction of these components is portability, modularity, and flexibility.

The C Language Integrated Production System (CLIPS) is used as the inferencing and control mechanism. The design methodology proposed is the Object Oriented Programming approach. The emphasis of this thesis is on interface tools and presentation systems that allow for linking and integration into the ITS shell proposed.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: [X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL: Yuh-jeng Lee	22b. TELEPHONE (Include Area Code): (408) 646-2361	22c. OFFICE SYMBOL: CS/Le

Approved for public release; distribution is unlimited.

Design of An Intelligent
Tutoring System Shell

by

Robert E. Scurlock Jr
Captain, United States Army
B.S., United States Military Academy, 1982

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

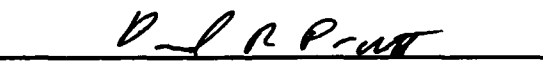
NAVAL POSTGRADUATE SCHOOL
September 1991

Author: 

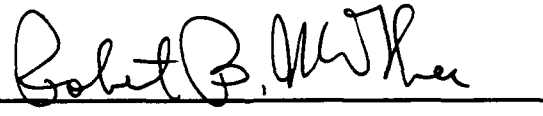
Robert Edward Scurlock Jr.

Approved by: 

Yuh-jeng Lee, Thesis Advisor



David Pratt, Second Reader



Robert B. McGhee, Chairman
Department of Computer Science

ABSTRACT

Computer technology has brought about numerous changes in the availability of educational media, especially the Intelligent Tutoring System (ITS). Since the development of an ITS is such an interdisciplinary task, the instructor needs assistance in developing these educational aides. An ITS shell, or authoring system, is the tool that will enable ITSs to make the transition from the research arena and into the educational environment.

The conceptual model of the ITS shell proposed in this thesis uses a layered approach to accessing the different modules of the ITS. The components, or subcomponents, of each module consist of either existing programs, or are selectable options developed by area experts. These options should allow the instructor to develop an ITS concentrating on the material being presented and on the method of interaction the student has with that material. The emphasis on the construction of these components is portability, modularity, and flexibility.

The C Language Integrated Production System (CLIPS) is used as the inferencing and control mechanism. The design methodology proposed is the Object Oriented Programming approach. The emphasis of this thesis is on interface tools and presentation systems that allow for linking and integration into the ITS shell proposed.

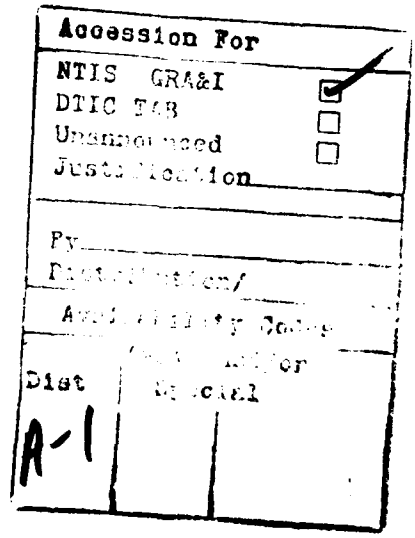


TABLE OF CONTENTS

I. INTRODUCTION

A. BACKGROUND

Many organizations are struggling to keep up with current technology and high personnel turnover rates. This is especially true for the military where continuity of skills and purpose are paramount. Additionally, the high cost, the destructive nature and potential safety risks presented by equipment with which military personnel are required to maintain proficiency, as well as administrative and routine tasks, highlights the need for alternate training methods. The military has become increasingly more dependent on simulations and simulators, but there is an additional resource that has yet to be fully exploited. That resource is the Intelligent Tutoring/Training System (ITS).

ITSs provide an innovative method to train and educate personnel by capitalizing on computer technology. For many years, Computer Aided Instruction (CAI) was nothing more than an electronic "page turner" that followed the same sequence of instruction no matter what the student/user's level of expertise. With the many advances in the field of Artificial Intelligence (AI), the "intelligent" component was added to form what has become known as Intelligent Computer Aided Instruction (ICAI), Intelligent Training Systems, or more commonly Intelligent Tutoring Systems. Moreover, ITSs have been defined as "...that field concerned with the application of artificial intelligence principles to the development of instructional programs." (AI Exchange, 1989, p. 6) In section II.B, we will review the design of ITSs and determine what makes them intelligent.

B. THE PROBLEM

ITSs have been developed using a variety of techniques. There is a great deal of research going on in this area to provide reliable, workable products that can be implemented in a wide range of different environments. Some major obstacles that delay practical applications of the technology have been identified. First, the majority of existing ITSs require large software development teams and special hardware to implement it. After this large investment of manpower and resources produced a finished product, it was limited to the single domain it was developed for and restricted to the complex platform it was implemented on. These research platforms are usually cost-prohibitive for most educational communities.

Second, most teachers have little or no experience with computers at all, much less the programming ability to develop a complex ITS program. An interesting fact was shown by Cable News Network (CNN) on a 7 June 1991 quiz. The results of a nationwide survey showed that fifty percent of all teachers had never used a computer. This survey showed that a large number of our educators have little or no computer experience. Those instructors who want to incorporate computer technology into their educational scheme most likely do not have the programming knowledge or the extensive amount of time to develop an ITS on their own. They need assistance in developing the advanced form of tutoring provided by an ITS. Therefore, by providing an ITS software development tool or an ITS shell, the instructor who wishes to incorporate computer

technology as an educational aide is much more likely to be able to incorporate these valuable tools in their teaching strategy.

Additionally, the design and development of an ITS incorporates the expertise of many research areas. To build an effective ITS requires input from computer scientists, psychologists, domain experts, educators, instructional designers, knowledge acquisition personnel, human factors engineers and cognitive scientists. Beverly Woolf argues, however, that building an ITS is not an application area where off-the-shelf material produced by other researchers can be used to build an ITS. (Woolf, 1988, p. 39) Perhaps not all ITSs are best produced in this manner, but an argument of this thesis is that it is possible to use properly tailored, generic components to construct an ITS. The time needed to produce an ITS could be greatly reduced if the different components of the ITS could be developed and coded by the area expert for that component. By providing an ITS shell and authoring tools, the instructor could take maximum advantage of existing technology and existing programs written to perform the required functions of the ITS components. This will allow the teacher to concentrate on the subject material without worrying about the programming aspect of the task.

Not all subject domains can be optimally implemented with a generic tool. However, the initial development of a working product can be optimized through a coordinated effort between the instructor and the programmer if the situation warrants. The more complex the domain and the stricter the hardware constraints, the more the programmer must be involved. The important point is to provide the instructor with the proper tools to produce his product.

One of the most difficult bottlenecks to overcome in the implementation of a new technology is the transfer of the engineering and production process out of the research arena and into the mainstream of industry (Pirolli, 1991, p. 107). In order for ITSs to make this transition from a research tool into the instructional environment, teachers will need the tools to create their own ITS. An ITS shell is the best way to help make this transition.

C. OBJECTIVES

In order to help make ITSs more accessible to and modifiable by instructors, the use of existing interface tools, presentation tools, and expert system shells can be incorporated into an ITS shell. The intent of this thesis is to provide a conceptual model of an ITS shell, and to demonstrate, through examples, how existing applications can be used to construct the various components of this shell. These components should stress modularity and portability. The use of the C Language Integrated Production System (CLIPS) is used as the central control element between components.

The main emphases of integration with existing systems are on the interaction between the student and the system and on the presentation of the subject material. The components should be developed by area experts. Once developed, a programmer will code the components for selection by the instructor. Most of these components will be briefly described and are left for examination in future works. The emphasis of these modular components should stress domain independence and should be configured to allow for instantiation in a domain and in a situationally dependent manner. This thesis

will take an in-depth look at the overall organization of the conceptual model, concentrating on the integration of the interface, the presentation tools, and the expert system control mechanism.

D. ORGANIZATION

Before going into detail on existing ITS work or into a conceptual model of an ITS shell, it is important to explain what is meant by an "intelligent" system and to explore the major components of an ITS. Although some in the field may use different terminology for these components, there appears to be general acceptance of what each module should contribute to the ITS. Figure 1 (Burns and Capps, 1988, p. 3) shows a fundamental ITS model that demonstrates how the student, the expert knowledge, and the interaction between the two, comprise the ITS. The four basic components of an ITS are an expert module, a student diagnostic module, an instructional module, and an intelligent interface. Also, since the development of ITS is such an interdisciplinary undertaking, the dimensions of communication, instruction, and expertise must be considered. (Burns and Parlett, 1991, pp. 2-3)

Chapter II gives a brief explanation of what constitutes an intelligent system and a brief overview of these basic modules of an ITS are given. Related works of other researchers that contributed to the formulation of the proposals and examinations of this thesis are presented in Chapter III. Chapter IV looks at the considerations of developing a shell. Chapter V details the conceptual model of the ITS shell and provides examples of the integration of system components. The conclusions are presented in Chapter VI

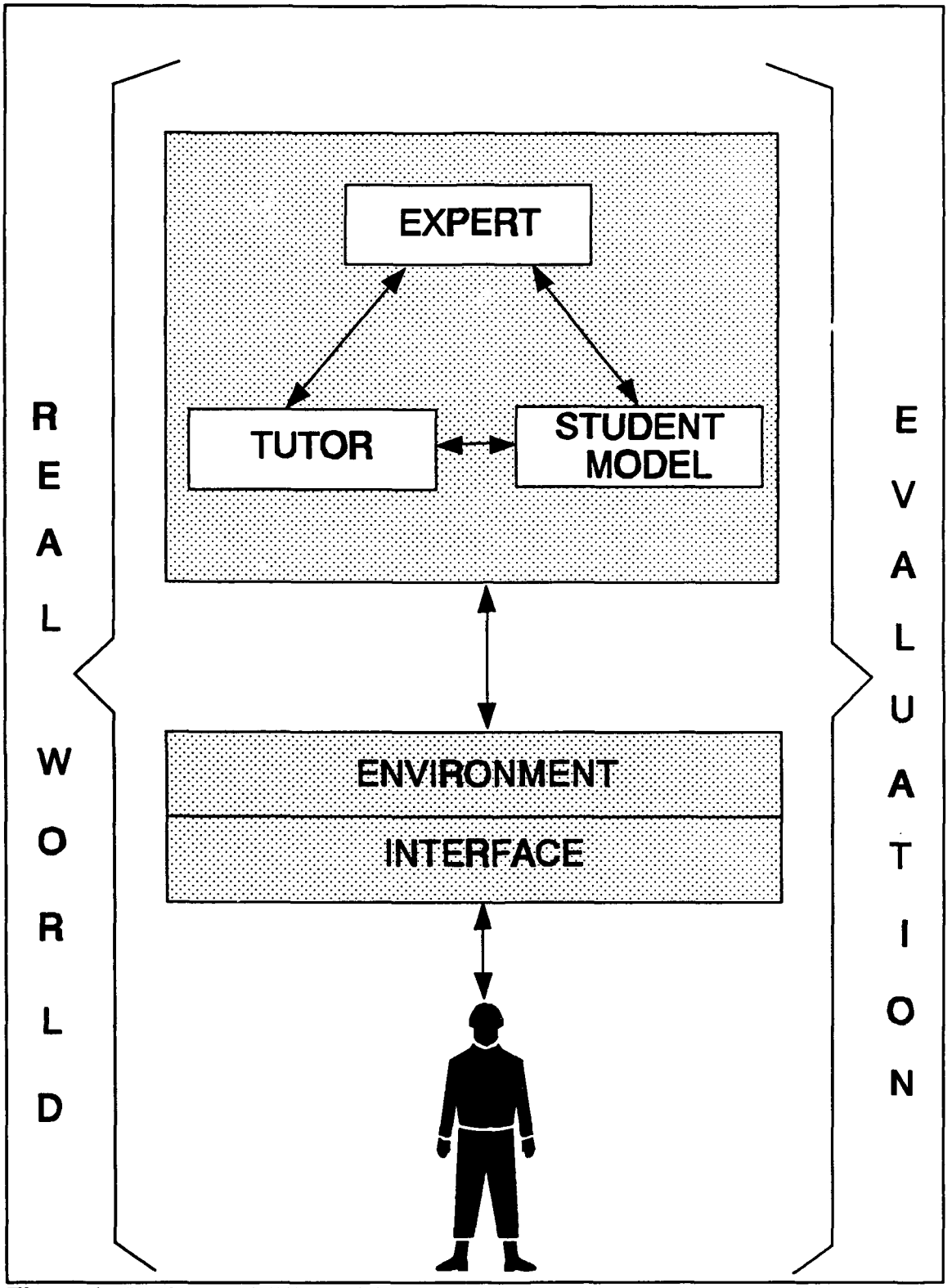


Figure 1 Fundamental ITS Model

detailing lessons learned and the future directions the development of this ITS shell could take.

II. OVERVIEW OF INTELLIGENT TUTORING SYSTEMS

A. INTRODUCTION

An intelligent system is one that is more flexible and adaptive than the traditional sequential computer program in that it is able to draw on knowledge and the power of association and inference to steer the running program toward useful results. To better understand this definition, it is necessary to define what data, information, and knowledge are. Data can be thought of as any value or entity that is available to the system for processing. Information is data that has been selected and organized for processing. Knowledge is information that is structured in such a way to bring out and exploit the relationships among the pieces of data. (Bielawski, 1991, p. 4) It is this knowledge that we wish to organize and present to the student in the most effective way in order to achieve our educational objectives.

First, an intelligent system should have the ability to use knowledge to complete a given task or to solve a problem. Second, the system should be able to exploit the powers of association and inference when trying to solve complex problems that resemble the real world. (Bielawski, 1991, p. 5) With respect to an ITS, the system should know where the individual student is in his current understanding of the instruction and how to progress.

There are certain characteristics we would expect an intelligent system to adhere to, including behaving logically, being responsive and adaptive, providing a nonlinear method

to navigate through the program and the knowledge domain, being able to use incomplete information by using existing information effectively, and most importantly, being user-friendly and be highly interactive (Bielawski, 1991, p. 6).

B. INTELLIGENT TUTORING SYSTEMS

ITSs have developed over the years, from the early 1970's to the present. Figure 2 (Redfield and Steuck, 1991, p. 280) shows what roles the different components played in the composition of four evolving ITS implementation theories. The control element was absorbed into the other components after the initial theory of 1973. The architectures remained basically the same with the shift being to the breaking up of the modules into subcomponents. (Redfield and Steuck, 1991, p. 280) These major components provide the basis for examining the functions required of the module and finding the best tool to achieve that function.

ITSs should be viewed as an additional media that the instructor can use to further a student's understanding of a particular domain. They should provide students with instruction that is tailored to the individual. This instruction should be conducted interactively with the student so that the student feels that the computer, or the ITS, is there to help her learn the material, and not just another presentation tool to flood her with more information. Some of the possible communication styles being examined to achieve this computer-student partnership in the learning process include didactic explanation, guided discovery learning, coaching or coaxing, and critiquing (Woolf and others, 1991, p. 74). If the system cannot help the student understand the material, they

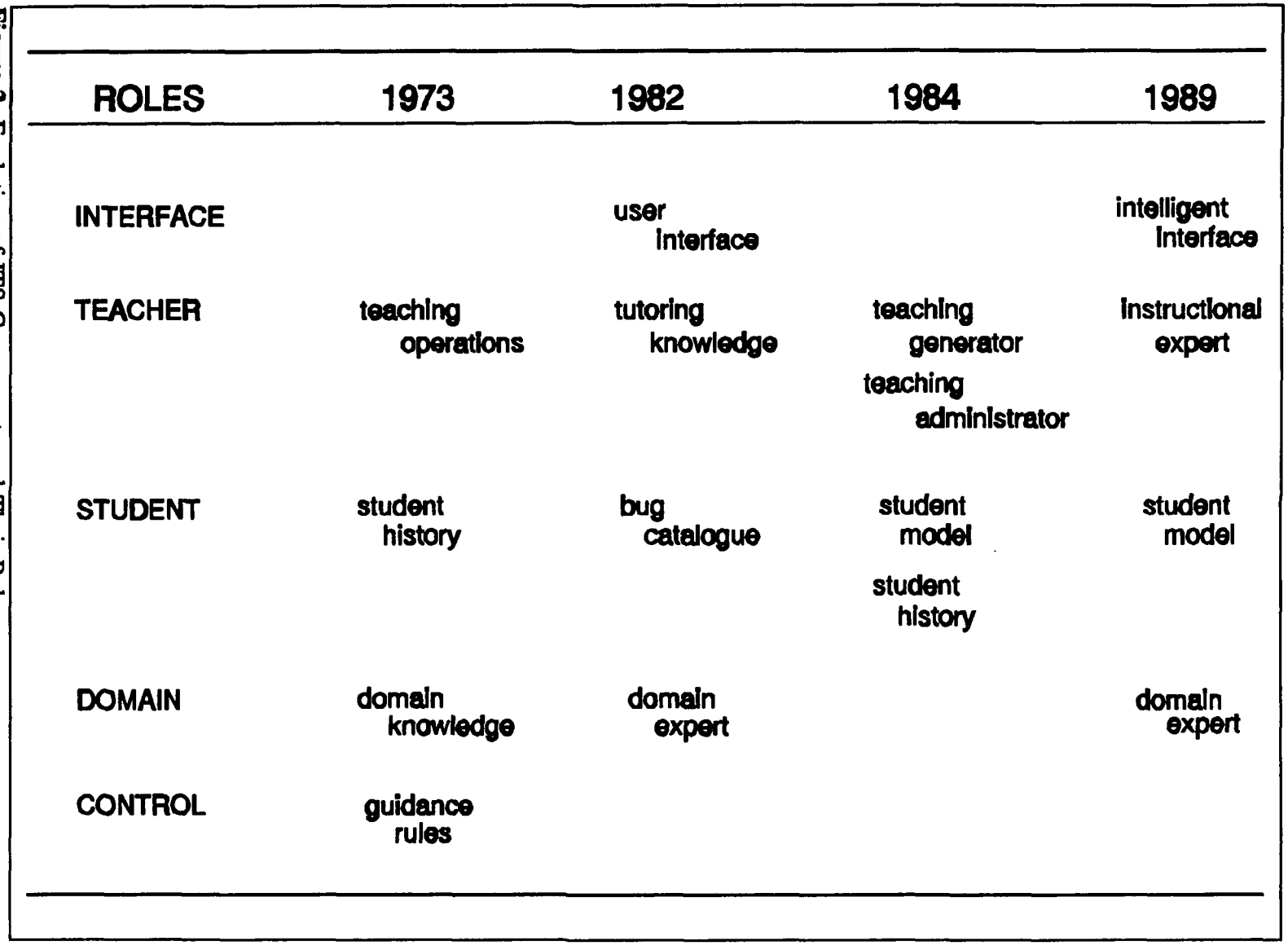


Figure 2 Evolution of ITS Components and Their Roles

could just as easily read a book or watch a movie about the topic. An ITS should "...provide a high-bandwidth method of user/system communication, infer student needs to redirect tutoring effort (based on a comparison of student performance with an internal model of domain expertise), and maintain separate knowledge bases for domain (subject matter) knowledge and pedagogical knowledge." (AI Exchange, 1989, p. 6)

This complex teaching strategy is incorporated into an ITS through the interaction of four main modules. Figure 3 (Burns and Parlett, 1991, p. 2) gives a simple anatomy of an ITS that was used to help classify the different areas for research. It is important to take a brief look at what each module is supposed to contribute to the instructional process, without going into great depth of how the module should be implemented. There are many articles and publications dedicated to just that purpose, and it is beyond the scope of this thesis to attempt to explore all the differing views. The next few sections will provide the foundational understanding of what an ITS should contain, although the complexity and the depth of incorporation will vary depending on system constraints and system goals.

C. THE EXPERT MODULE

The expert module contains the domain knowledge for the system (Burns and Capps, 1988, p. 2). It is the most difficult to develop of all the components. It is also the most critical, since a system is only as good and credible as the knowledge used by the system. The major bottleneck of most expert systems is knowledge acquisition and knowledge representation (Anderson, 1988, p. 22). The same is true for ITSs. There are

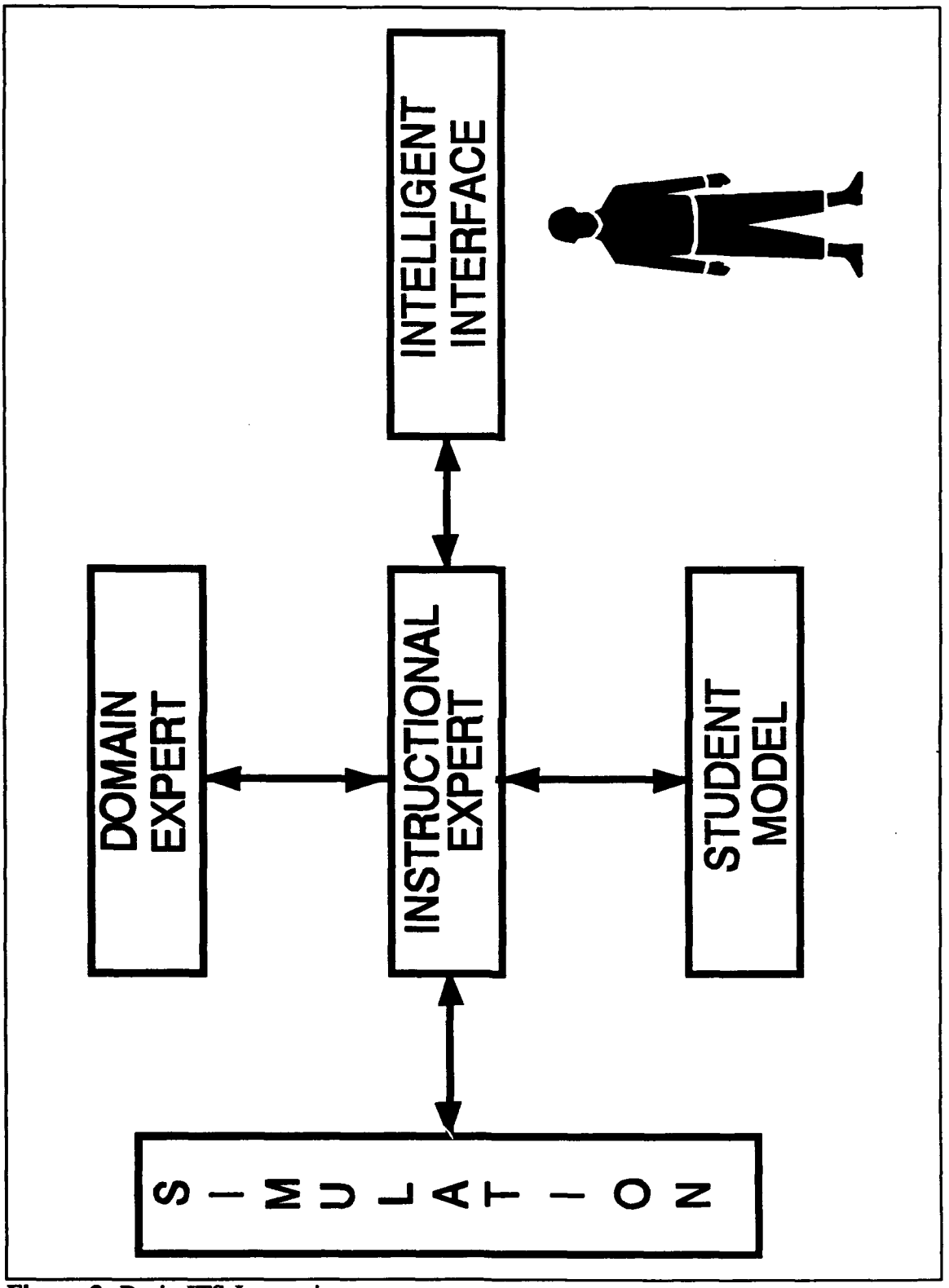


Figure 3 Basic ITS Interaction

many commercial tools on the market that have greatly assisted in the reduction of this stumbling block.

Although it is possible for a knowledge engineer to laboriously proceed through the knowledge acquisition process and spend numerous man-years producing a knowledge base for a single domain, it would be much more productive to provide the instructor with the tools to encode the knowledge himself. An even better solution would be to provide the instructor with encoded knowledge and allow him to augment that knowledge base with presentation material, such as hypermedia products. One such attempt at encoding a large knowledge base is the Cyc project. The Cyc project is an attempt to construct a knowledge base, consisting of approximately 200 million rules, that would cover the spectrum of human consensus knowledge (Lenat, 1990, p. 30). The reuse of existing knowledge bases is an area in AI that has been hotly debated and researched over the past decade or so.

Knowledge representation is a key issue in producing this module. It is this representation that determines the methods of presentation and the instructional strategies available for use in the ITS. Procedural knowledge is usually coded as rules and it explains how to perform a given task. Declarative knowledge is usually coded as facts and it states a fact about an object or topic. (Burns and Capps, 1988, p. 5) A third type of representation, and probably the most difficult, is qualitative knowledge. "...qualitative knowledge is the causal understanding that allows a human to reason about behavior using mental models of systems." (Burns and Capps, 1988, p. 5) The concentration in this thesis will be on procedural and declarative knowledge representation schemes.

The instructor need not be concerned with understanding the different types of knowledge representations available to him. The presentation tools provided in the shell should account for the selection of the appropriate representation method. As new object oriented presentation tools have become available, the differentiation between representations appears to have blurred. Hypertext and hypermedia tools now use pointers to presentation objects. These objects can be text, graphics, video, audio, or interactive voice depending on the node pointed to. (Bielawski, 1991, pp. 40-51)

D. THE STUDENT DIAGNOSTIC MODULE

The student diagnostic module represents the student's current state of understanding about the domain and uses the individualized model to tailor the instruction to the student's needs. The student model is a data structure describing the student's knowledge, and the diagnosis is a process that manipulates that data. (VanLehn, 1988, p. 55) There are many ways to determine the student's entry level knowledge and to update that model as the student progresses through the instruction.

The most common uses of the student model are to provide for advancement, to determine when to offer unsolicited advice, for problem selection and presentation, and for adapting explanations (VanLehn, 1988, pp. 56-57). This model can be in the form of a data base which stores pertinent facts about the individual student. As the fact base is updated by the student's progression, evaluations, or other criteria outlined by the instructor or the programmer, the instructional material and teaching strategies can be adjusted by rule firings controlled by the inference engine.

E. THE INSTRUCTIONAL MODULE

The instructional module is responsible for presenting the material to be taught in a logical manner in keeping with the student's level of progression. It also determines which teaching strategy to employ for each individual student at the various levels of progression through the curriculum. In other words, the primary function of this module is to provide the proper level of instruction, using the teaching strategy that is deemed correct for each student, at the proper point in the program of instruction. (O'Neil and others, 1991, pp. 69-83)

Cognitive scientists and behavioral psychologists have worked for many years to determine the optimal way to assist students through the learning process. In (Bower and Hilgard, 1981, p. 566) B. F. Skinner, a famous behavioral psychologist, proposed that a good tutor:

1. begins where the student is, and does not insist on moving beyond what the student can comprehend.
2. moves at a rate that is consistent with the ability of a student to learn.
3. does not permit false answers to remain uncorrected
4. does not lecture; instead, by his hints and questioning he helps the student to find and state answers for himself.

An interesting example of an instructional design methodology using a frame hierarchy is shown in Figure 4 (Woolf, 1991, p. 133). This designed network uses knowledge unit frames to build the relationships between topics, but does not restrict the way a student could traverse this network. The elements of this network describe an

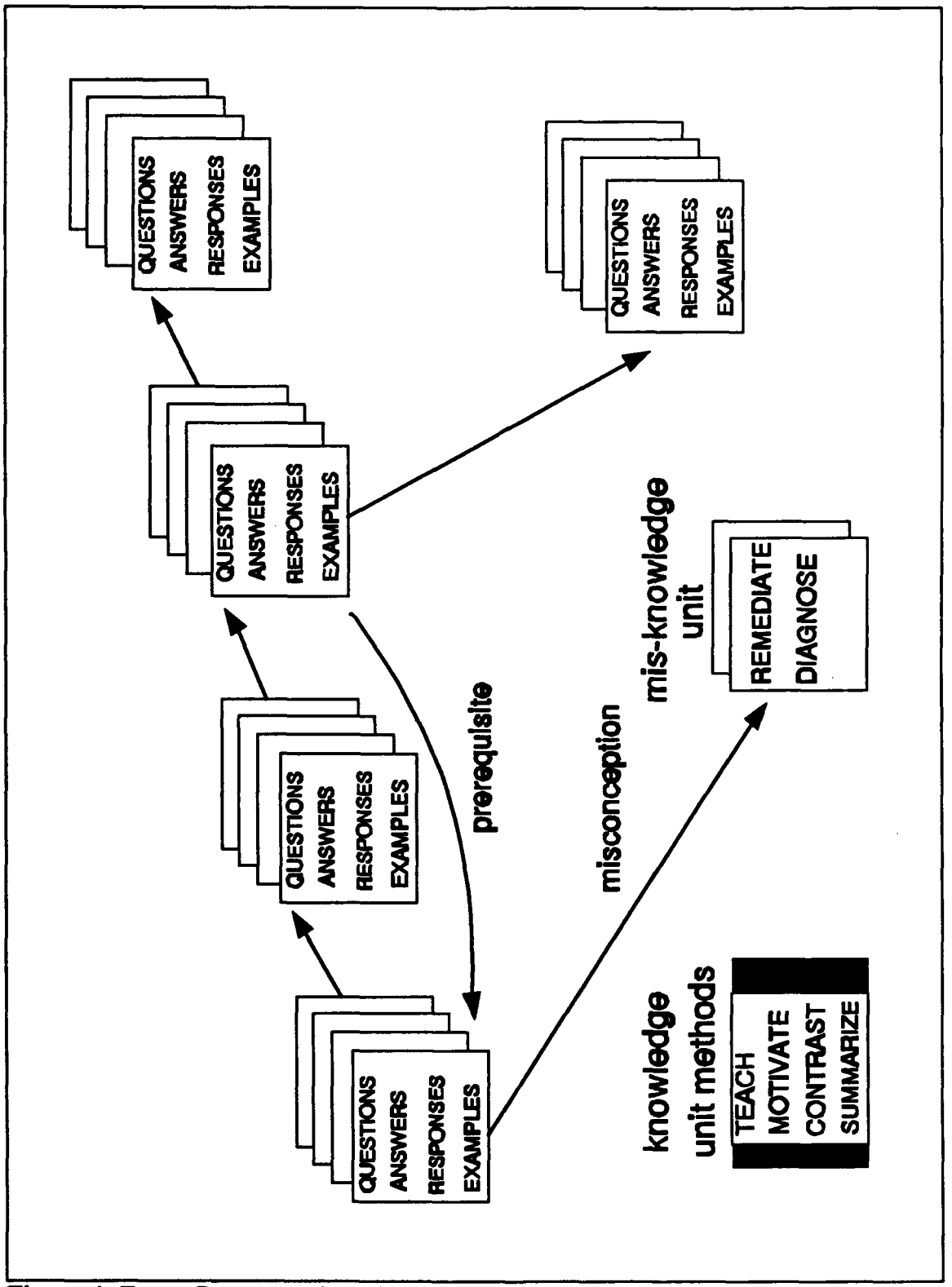


Figure 4 Frame Representation of Instruction

attempt at representing and reasoning about tutoring primitives. (Woolf, 1991, pp. 128-136) This hierarchy lends itself well to an object oriented approach and demonstrates one of many methods of controlling a tutoring session.

F. THE INTELLIGENT INTERFACE

The interface is the key element in forming a interactive, flexible learning tool. The interface is the tool that allows the student to interact with and explore the expert knowledge in the ITS. It must be kept simple and it's use must be intuitive to the student. A good evaluation is that an ideal interface for an ITS should present "...a conceptual vocabulary, tied to standard notational conventions, with metaphors, pictures, and labels that tie the vocabulary elements to applicable situations in the world." (Bonar, 1991, p. 46) The closer the simulation or instruction is controlled in the ITS, to the way it would be controlled in the real world, the more effective the system will be. The interface should also be designed to minimize the student's interaction with the computer itself, and allow the student to concentrate on the instructional material.

There are tradeoffs in the interface design, too. The closer the interface is linked to knowledge base the more system dependent the ITS becomes. Also, the more elaborate the interface, the greater the memory requirements and the more demanding the system will be on the hardware to produce a real-time simulation.

III. AN EXAMINATION OF TUTORING SYSTEMS

A. INTRODUCTION

A great deal of research effort is being conducted trying to realize the full potential and optimism that has surrounded ITSs. Many of these projects are restricted to a limited domain and specialized machines. Much can be learned from the contributions of these projects, but the main focus of this thesis is to produce an ITS shell design that is not proprietary and that is highly portable, flexible and easily upgraded. The expert system shell that provides this flexibility is CLIPS.

Many research efforts have advanced the current state of design and development of ITSs. It is important to look at a very small sample of these works to show what influenced the design ideas discussed in the next chapter. Importance was placed on models that provide the most flexibility, portability, and the best integration with other languages and existing applications. Having stated above the selection of CLIPS as the expert system shell, much of the research effort went into gaining a better understanding of the capabilities of CLIPS and examining its use in a variety of applications.

B. A MODEL-BASED GENERIC TRAINING SYSTEM

The first example is a model-based ITS for a Generic Training System (GTS) for industrial use. Figure 5 (Inui and others, 1989, p. 60) shows the architecture of this system. This system uses the tools in expert systems that handles knowledge as data so

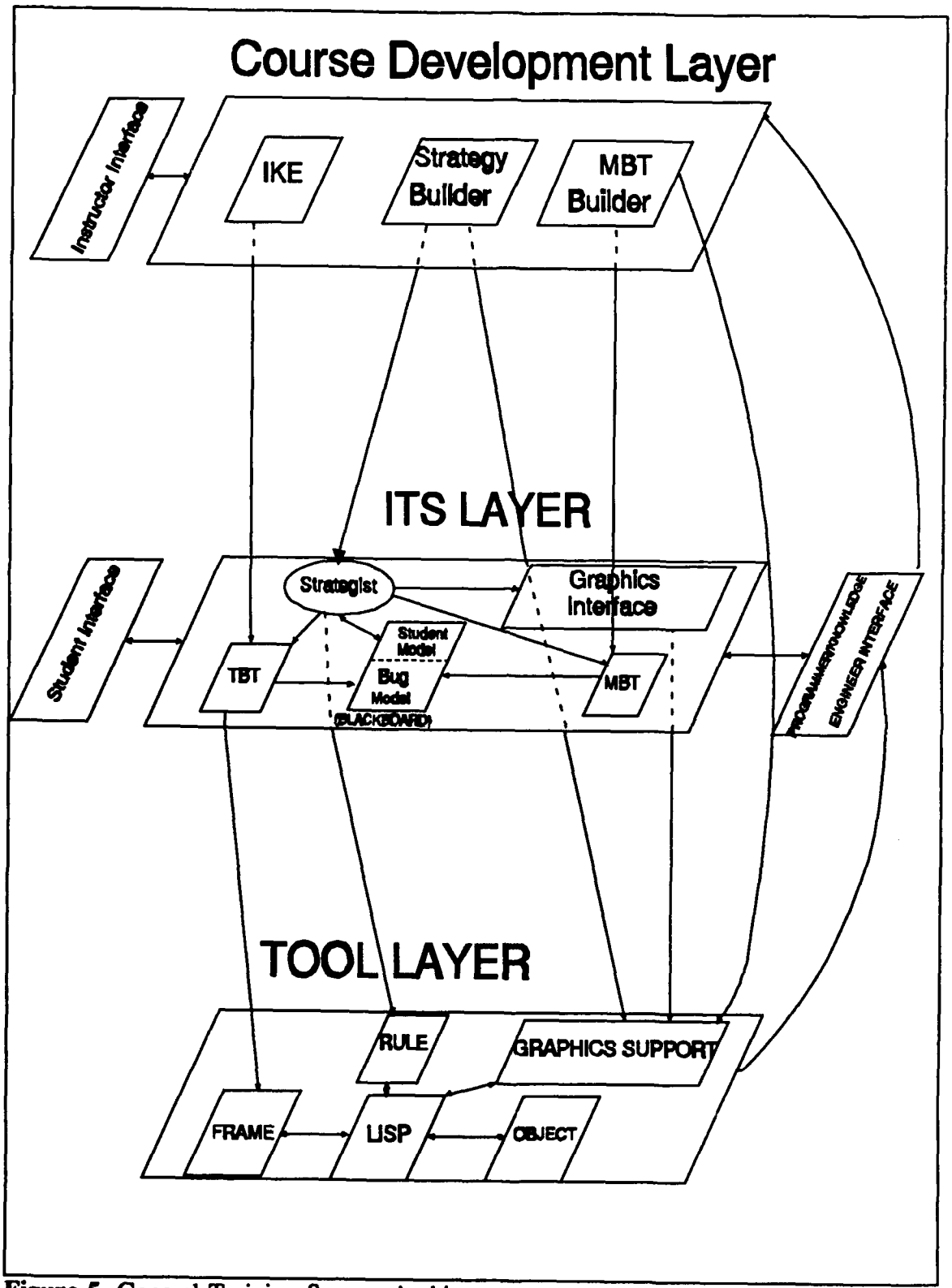


Figure 5 General Training System Architecture

it can be completely separated from the reasoning system, making it easy to replace knowledge. The knowledge of the specific domain is separated from the training methods by using rule bases, frames and object oriented programming language construction. (Inui and others, 1989, p. 66)

The GTS uses FRANZ LISP, OPS5, Package for Efficient Access to Representations in LISP (PEARL) and Flavors programming languages and tools to implement the various layers. Although these are very powerful tools and the structure appears very sound, this architecture is limited to high-end workstations for implementation. Many of the basic ideas of this model were used to form the basis of the generic shell describe in later chapters. The tools examined for the implementation of this shell are quite different, however.

C. PAYLOAD-ASSIST MODULE DEPLOYS/INTELLIGENT COMPUTER-AIDED TRAINING SYSTEM

The Payload-assist module Deploys/Intelligent Computer-Aided Training (PD/ICAT) system is an ITS that was developed on a LISP machine and transferred to a UNIX workstation. The initial intent was to translate the LISP code and Automated Reasoning Tool (ART) rule base into C and CLIPS for better portability and availability of hardware and software platforms that the system could run on. The secondary goal was to develop a generic architecture and general purpose development environment that could be used to produce other ICATs more rapidly. (Hua, 1990, p. 69)

The first objective was achieved as the PD/ICAT was re-coded to operate on a UNIX workstation with an X Window interface. The general architecture developed is as shown in Figure 6 (Hua, 1990, p. 70). The goal of producing a general purpose ICAT development environment was not achieved. The major stumbling block, as is the case in most expert systems or ITSs, was in knowledge acquisition and knowledge representation. (Hua, 1990. p. 74)

This system proposed a different architecture but the modules performed the same basic functions as the architecture described in Chapter II. The idea of using a blackboard to interface between modules and the user provided the greatest contribution to the model explained in Chapter IV.

D. CLIPS ITS

The CLIPS ITS is an ITS that was developed solely using CLIPS 4.3. It is strictly text based and was developed to assist users in learning the basics of the CLIPS expert system authoring tool. CLIPS ITS demonstrates a simple approach to developing an ITS over a limited domain and the use of modular design of material being presented to optimize the use of limited memory space. The system is currently configured for use on a PC only. In order to work on other platforms, minor changes in the system calls and the interface used would have to be made. It incorporates a very limited student model and teaching strategy.

The major contribution of this system is to demonstrate the feasibility of coding any portion of an ITS in CLIPS and the use of modular presentation material. It also provides

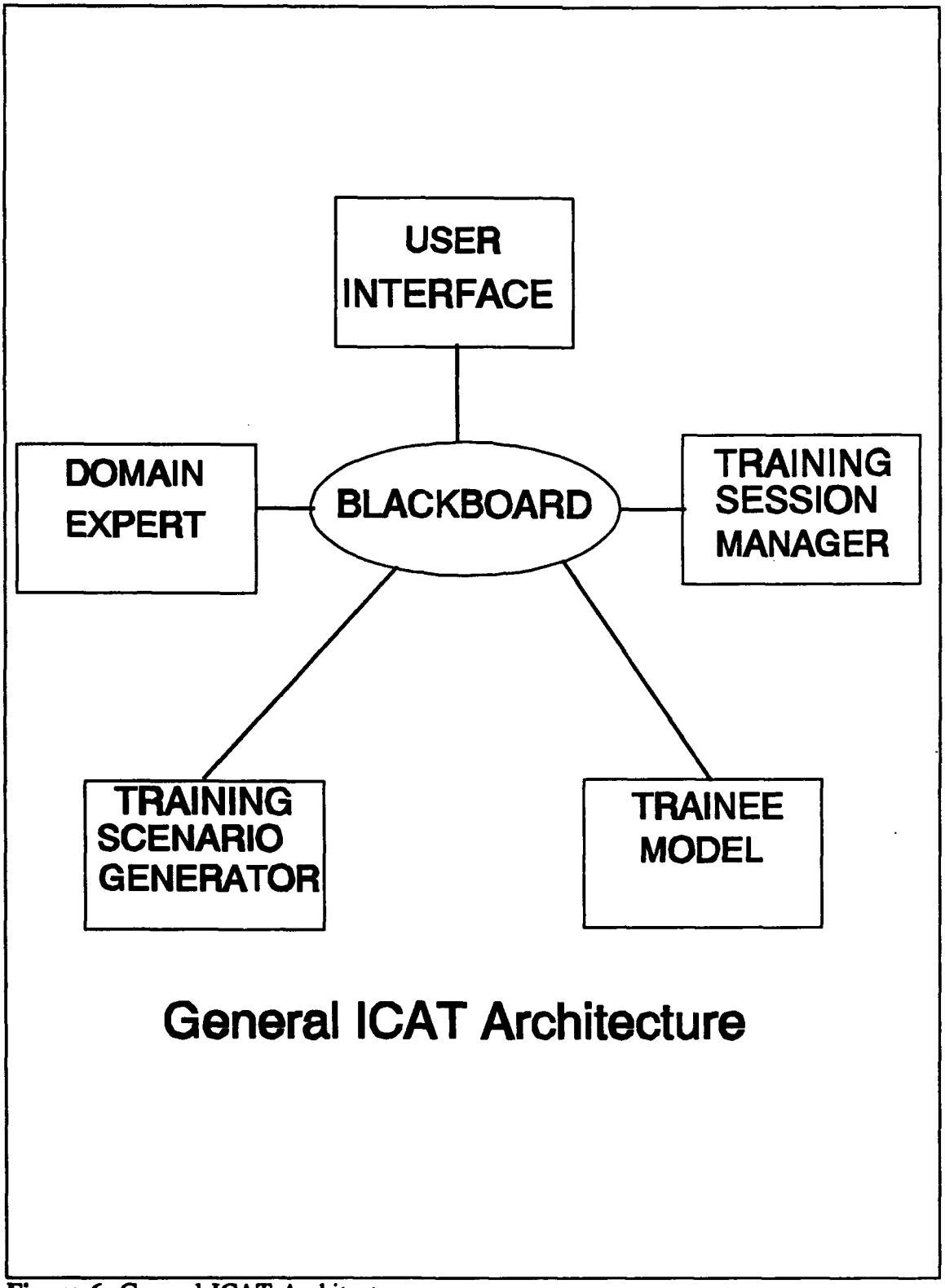


Figure 6 General ICAT Architecture

a good example of a basic student model using a CLIPS fact base. The use of CLIPS as the controlling agent for all components of the system is expanded as the controller for the system proposed in Chapter IV.

E. A MACINTOSH ITS THAT USES CLIPS

An ITS using CLIPS that was developed for the Macintosh using HyperCLIPS demonstrates the versatility of CLIPS on different platforms. The system effectively demonstrated how to pass control back and forth between the HyperCard interface and CLIPS. HyperCLIPS combines HyperCard, from Apple Computer, with CLIPS. HyperCard is a popular hypertext system which is used to build user interfaces to databases and other applications. The Map Symbol Recognition Tutor (MSRT), is a system used to instruct students on the skills of map reading. It was developed more to show the feasibility of building a flexible, easily adaptable ITS incorporating CLIPS with the graphical interface building capability of HyperCard. (Hill and Pickering, 1990, pp. 62-68)

A useful experiment to demonstrate the true flexibility of MSRT would be to transfer the CLIPS components to a PC or a UNIX system and create another interface for the given system, to determine the reusability of the domain knowledge being taught. For example, transporting the ITS to a PC and using Asymetrix ToolBook as the interface. This system demonstrates the ease of integration of a system with CLIPS and the system dependency created by the user interface.

F. AN EXAMPLE OF A HARD-CODED ITS

An Aircraft Recognition Tutor (ART) was developed using the Object Oriented Programming (OOP) paradigm. Figure 7 (Campbell, 1990, p. 34) shows the hierarchical structure of ART. ART shows how effective the OOP paradigm can be in implementing a system on a baseline PC-AT compatible 80286 machine with limited hardware and software requirements. The language used in the development was Turbo Pascal 5.5 which restricts portability of this system. The object oriented design, however provided great flexibility in modifying the system to be used to train students over other similar domains. (Campbell, 1990)

ART lacked the incorporation of an inference engine and had limited application of a student model. By using a procedural language alone, the system was unable to incorporate some of the more advanced approaches to implementing the different components of the ITS architecture. It also limited the incorporation of advanced tools that have been developed since ART was produced. ART is a good example, though, of a working product in the field using very limited hardware and software support.

In the experimentation phase of this thesis, a number of the instructional sets presented in ART were created using an existing presentation tool. The topics were presented after a rule in CLIPS was fired. The intent was to show the feasibility of creating modular components of the topics to be presented and controlling them with CLIPS.

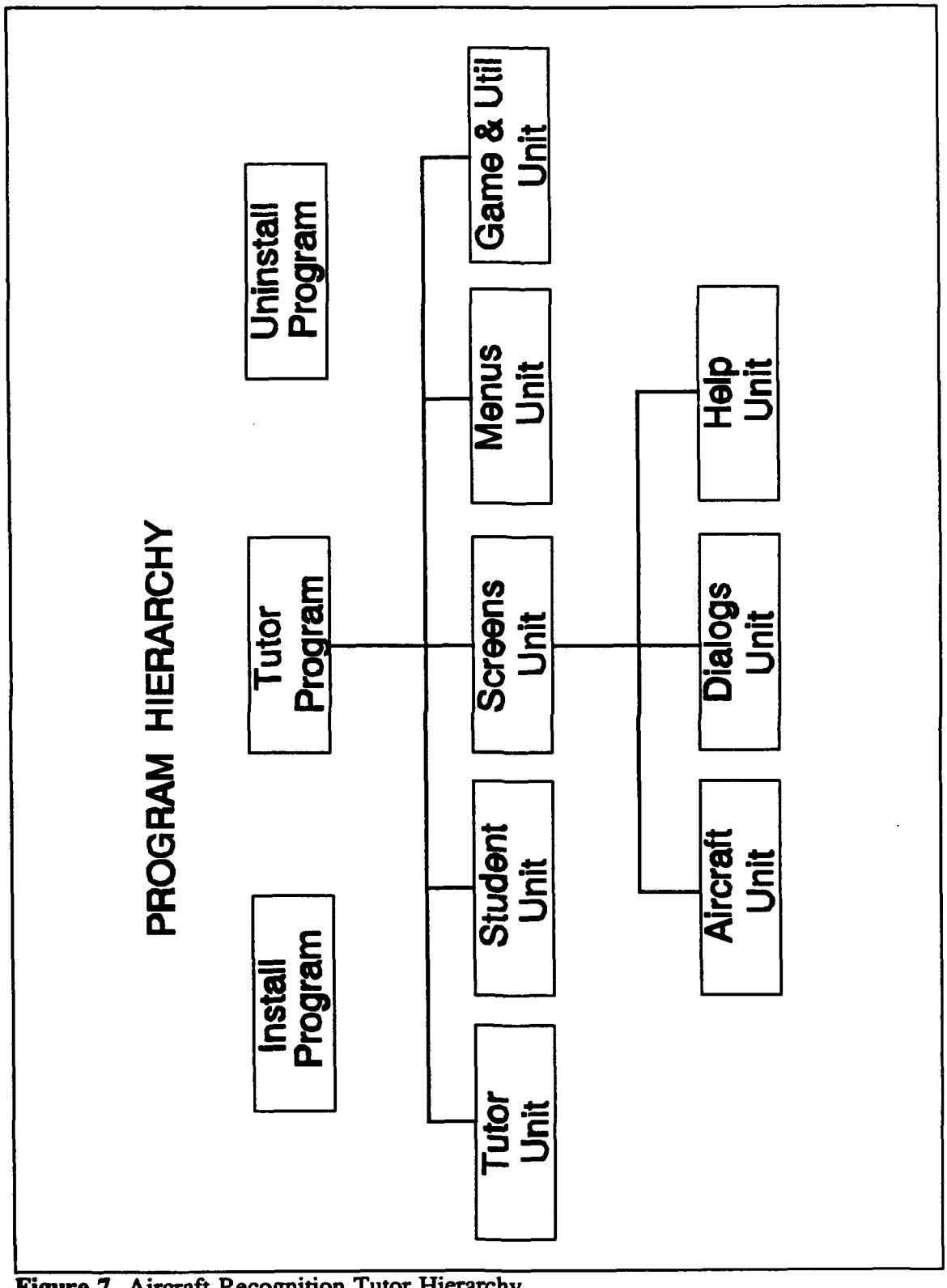


Figure 7 Aircraft Recognition Tutor Hierarchy

IV. DEVELOPING A SHELL

A. INTRODUCTION

The main objective in the development of a generic ITS shell is to remove the burden of integration and development of the system from the educator. The goal is to allow the educator to concentrate on the subject material to present and the methods of interaction that he wishes to choose. To achieve this goal, the area experts would develop the code for specific components of each module and the programmer would either provide the resources to conduct the linking of these components or he would handle this linking process himself.

The shell should provide the instructor with the necessary tools, for his individual computing environment, to select and modify components for the necessary modules of the ITS. To achieve this goal the shell must have a modular design and provide for incremental development of the system. This design should also be geared toward a high level of portability to allow for reuse of individual components on other platforms or in other systems.

Many tools that would provide for this type of modularity and integration are either new or are still in the research stages. Some of the considerations of methodologies to be explored and systems that could possibly be used will be discussed in later sections. The use of an object oriented programming approach, the connection of existing applications, and the use of blackboard systems are considerations to achieve these goals.

Also, the expert system shell that provides the goals of portability and easy of integration is CLIPS.

B. THE OBJECT ORIENTED PROGRAMMING PARADIGM

Object oriented programming (OOP) is a relatively new approach that is geared at more closely modeling the real world at a high level of abstraction. It allows the programmer to describe objects and their real world behaviors rather than defining data formats and procedures and subroutines to manipulate that data. (Elliot, 1990, p. 20) A very general definition of object orientation is "...the software modeling and development (engineering) disciplines that make it easy to construct complex systems from individual components." (Khoshafian and Abnous, 1991, p. 6) Since there have been many books and articles written attempting to explain what OOP is and how it works, the concentration in this section is how it is suited for constructing a shell.

There are five features that are generally accepted features of OOP languages. The first feature is inheritance. A class is a "...template which describes the common characteristics or attributes of objects." (Giarrantano, 1991, p. 4) Classes are arranged in a hierarchy which allows classes below, the more specific classes, to inherit attributes and message handlers from the classes above, the more general classes. This allows the programmer to construct reusable code through the use of inheritance. (Giarrantano, 1991, pp. 3-5)

Another benefit is encapsulation of data within an object. Encapsulation is the protection of the attributes and message handlers that define the given object. The

attributes cannot normally be changed or affected unless a message is specifically sent to that given object. This helps eliminate side-effects. (Giarrantano, 1991, pp. 3-5)

The other three features are abstraction, polymorphism, and dynamic binding. Abstraction is the use of an object to "...describe a real-world object or system...." that is being modeled (Giarrantano, 1991, pp. 4-5). Polymorphism is "...the ability of different types of objects to respond differently to the same message type...." (Giarrantano, 1991, pp. 86-87) Dynamic binding is the ability to assign the object reference, or name to different objects at run-time. This allows flexibility in programming since the exact target object may not be known in advance. (Giarrantano, 1991, pp. 86-92)

These definitions and features are only intended to demonstrate the power of OOP. OOP is being called the programming methodology of the 1990s because it provides a better way for:

1. Modeling the real world as close to a user's perspective as possible.
2. Interacting easily with a computational environment, using familiar metaphors.
3. Constructing reusable software components and easily extensible libraries of software modules.
4. Easily modifying and extending implementations of components without having to recode everything from scratch. (Khoshafian and Abnous, 1991, p. 1)

These benefits of OOP are key elements needed to provide the modular, portable components of the ITS shell described in Chapter V.

C. PROGRAMS THAT CALL OTHER PROGRAMS

Different programs perform specialized functions that can uniquely perform the required function of a component, or subcomponent, of a module in the ITS shell. In order to integrate these different programs into the shell, they must be able to interact

with one another. This means a program must be able to pass information to another program, temporarily relinquishing control to it while it performs the required task. Once completed, control must revert to the next required program to continue the instructional process.

CLIPS allows for external calls to other programs. This ability was successfully tested as part of this thesis to call presentation programs. The linking of programs can be done either by these external calls, or CLIPS allows for embedding a program within another. These considerations are system dependent, as resource requirements will determine the feasibility of the method. Memory constraints and processing time required to transfer control are prime considerations. Tools are available to allow for connecting programs in a multitasking environment, too.

One such tool was developed by the Microsoft Corporation. Dynamic data exchange (DDE) is a Microsoft Windows protocol that allows applications to continually share information between other running applications (Borland, 1991, pp. 192-193). The concept can be effectively compared to passing information between components of a shell by placing information to be processed by one component onto a clipboard, and calling the other component to process that data. Once the required action is complete, control is returned to the controlling element, which invokes the next required application or component.

DDE is a new protocol for linking applications on a PC under Microsoft Windows, so many new applications are in development. One example of applications linking using DDE, discussed in Section V.C.1, is between Asymetrix's Toolbook and Microsoft's beta

version of Multimedia Development Kit. As more systems are produced this method of linking will provide a better evaluation for incorporation into an ITS.

D. BLACKBOARD SYSTEMS

Blackboard systems are another method available to solve problems using multiple resources. A blackboard is divided into the knowledge sources and the blackboard data structure. The knowledge is broken up into modules, each with it's own inference engine. These modules work together by placing information into working memory, in a format the intended module can read. There is essentially no control element in the system, as the knowledge sources in the system dictates the system's actions. (Engelmore and Morgan, 1988, pp. 1-10)

As the modules contribute to the solution of the problem, the partial, or working solution is stored in a solution space data structure. The problem spaces are arranged in a hierarchy to provide the proper sequence to complete the solution. Each solution space is updated based on it's current state and the knowledge provided to it from the different knowledge bases. (Engelmore and Morgan, 1988, pp. 4-14)

Blackboard architectures are a relatively new approach to problem-solving. Since a blackboard system was not available for the PC, the baseline system used for this thesis, the implementation of an ITS shell using this architecture is left for later research. The theoretical design and operation of such a system seems well suited for this type of application, however.

E. CLIPS AS THE CONTROL ELEMENT

CLIPS was developed by NASA to overcome operational computing constraints, which was traced to their use of LISP-based tools, particularly because of the low availability of LISP on a wide variety of conventional computers, the high cost of state-of-the-art LISP tools and hardware, and the poor integration of LISP with other languages. CLIPS is written in C to support the goals of high portability, low cost, and ease of integration with external systems. There is also a version written in ADA which is only available on UNIX based systems. CLIPS 5.0 also comes with the CLIPS Object Oriented Language (COOL) which was developed by combining features and functionality of Common Lisp Object System (CLOS) and Smalltalk. Therefore, CLIPS provides for both procedural programming and object oriented programming approaches. (CLIPS Reference Manual, 1991, p. xiii)

CLIPS is a powerful expert system shell that is provided to government agencies and government contractor for the exchange of the required media, and can be purchased for commercial use at low cost (CLIPS Reference Manual, 1991, p. 217). CLIPS comes with all the source code for all modules so it can be tailored for specialized applications. Additionally, CLIPS has been tested and used on the IBM Personal Computer (PC), Macintosh, and the CRAY and numerous machines in between. (CLIPS Reference Manual, 1991, p. ix) It is a very versatile tool and can effectively provide the inferencing capability and the rule-based control to construct intelligent systems.

The knowledge representation in CLIPS is provided in the form of rules, facts, and objects.

> Rules are referred to as procedural knowledge (they tell us how to proceed to change states) and facts are declarative knowledge (they describe the state of the system at any point in time). Together, the procedural and declarative knowledge are referred to as a knowledge base. (Baudendistel, 1990, p. 3)

With the new feature of COOL, objects and message handlers are also a means of developing elements for the knowledge base. The use of message handlers to cause certain actions to occur can greatly enhance portability. For example, a message sent to an object telling it to show a presentation is converted to the action of showing the presentation for that object. The same show message can work for many objects and can be adjusted easily by the programmer to adapt to the hardware configuration and presentation tool available to the given system.

V. THE CONCEPTUAL MODEL OF THE SYSTEM ARCHITECTURE

A. INTRODUCTION

There is no doubt a need for authoring tools and shells to assist in the development of ITSs. A major point to keep in mind when developing a system is that without credible knowledge in the knowledge base and without a sound instructional design, there is no tool that can produce an effective learning system. Therefore, any tool incorporated into a component of the shell must enforce good, logical design criteria and assist the instructor in preparing an effective instructional design. The expert system shell should also incorporate truth maintenance capabilities to assist in avoiding knowledge conflicts. CLIPS 5.0 allows for truth maintenance incorporation, which reinforces it's selection for this shell.

The ITS shell should not just allow an instructor to construct a system, it should also provide guidance and reasonable default settings that are properly suited for the needs of that system configuration. For example, the teaching strategy should be geared toward the type of knowledge being presented and the student's level of understanding. Therefore, the shell should adjust the default settings and provide recommendations to the instructor about the available choices. So, the shell should not only make the various tools available to the instructor, but also provide help and guidance about which option to use based on previous choices and system constraints.

The basic architecture of the model proposed in this thesis, shown in Figure 8, takes a layered approach. The user should be able to interactively control his exploration of the knowledge domain within the constraints established by the instructor. The instructor should be able to choose and modify the selectable components without corrupting the baseline code or application. He should also be able to test and evaluate his system throughout the incremental development as if he were the student. In other words, the user at a given level has control over the functions at his level and all those levels above him, but he cannot change anything below his level.

In order to be a workable tool, the shell must be clearly understandable so the user can spend his time and energy using the system. The time and energy expended trying to master a complex tool must be minimized. This puts the burden on the programmer to set up the system options and interface such that the instructor can readily identify how to control and manipulate the system. He should also provide solid examples and detailed help functions should the instructor need them.

As products are integrated into the shell, there must be a clear understanding that there is many trade-offs between ease of use, generality, and resource requirements. The more elaborate the tool, the more expensive the cost of resource usage. For example, extensive graphics integration in an interface or presentation is much more memory and computationally intensive and takes considerable more display time than a text-based system. Therefore, the instructor may not always be able to incorporate the more elaborate components, such as natural language interfaces, or video disk presentations if the system hardware will not support it.

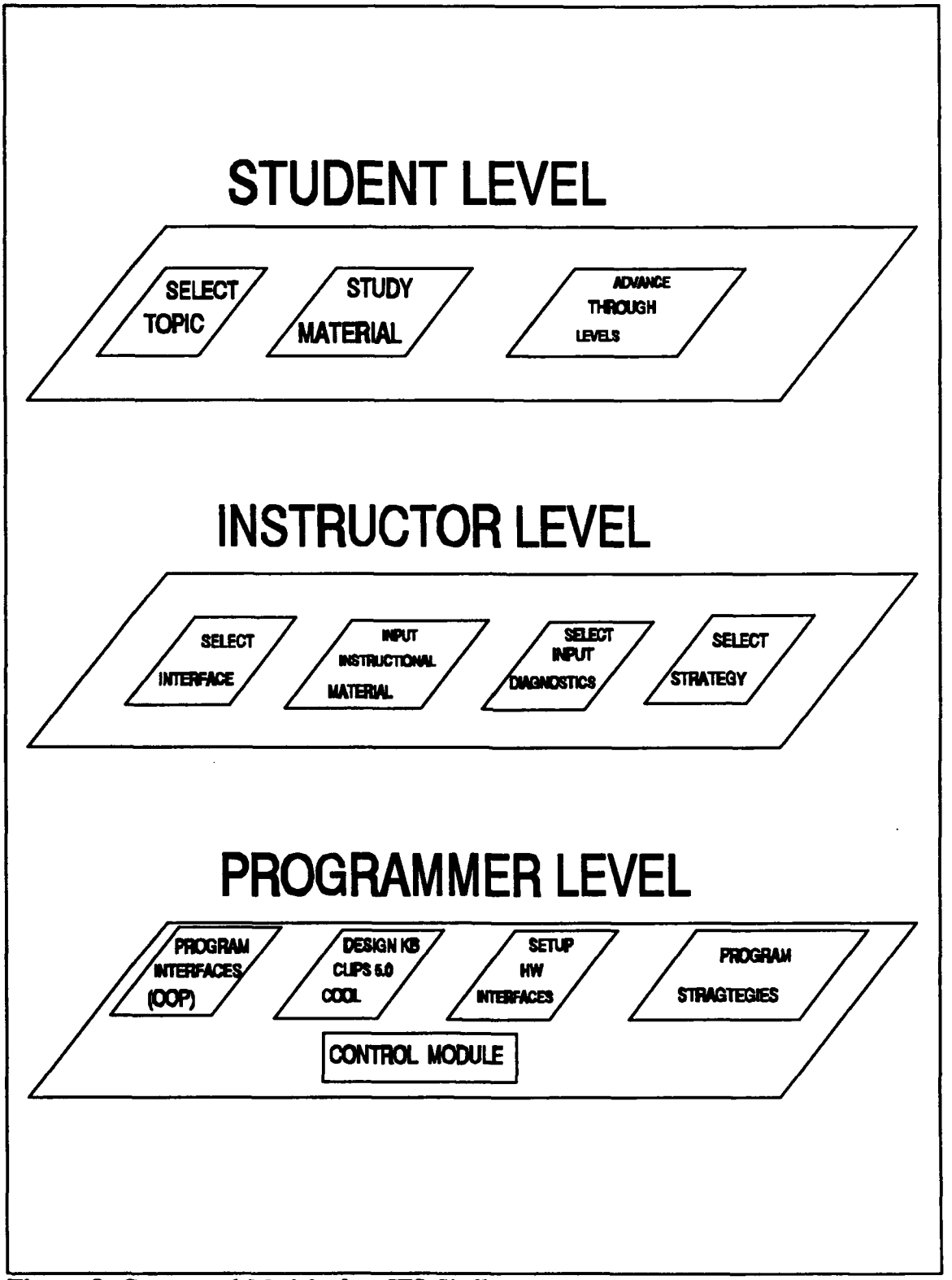


Figure 8 Conceptual Model of an ITS Shell

These resource problems are being resolved as technology continues to improve and new solutions to memory restrictions are developed. For example, the new CLIPS version 5.1, which according to Brian Donnell of NASA's CLIPS Development Team, will be available in a Microsoft's Windows configuration. This new version is due out in November 1991. (Donnell, 1991) This would allow the incorporation of multitasking and DDE links discussed earlier in section IV.C.

The use of OOP and modular design for the shell is intended to enable easy incorporation of new technology. As new hardware increases system capabilities, or new software developments improve applications, the shell components should be easily upgraded. This allows the system to keep pace with available technology, without having to completely redo the entire system. Only the components that need to be changed will be redone. This will save many man-hours by reusing the existing components.

B. SELECTING AN INTERFACE

The interface is the component that will determine how well the student will be able to navigate the intended instructional material. This is where ease of use is paramount. The user must clearly understand the operation and manipulation of the system so he can concentrate on what the system is trying to teach him, and not spend unneeded effort and frustration with the system itself. The interface should be the starting point in order to achieve the goal of building a flexible tutoring environment. The system is then built behind the interface. This supports the current research theories that ITS design "...is

moving toward student-centered, reactive learning environments." (Burns and Parlett, 1991, p. 3)

There are many existing interface tools that have been developed, covering a wide range of complexity and cost. From simple text-based windowing tools to natural language processing systems, these tools provide the user with the means to interact with the system. The design of the interface is more geared to how to do something than it is to what things to do (Burns and Parlett, 1991, p. 2). The determination of what to do will be handled by the other components. Some of the interface tools examined in this thesis are described in the following sections.

1. Windowing Tools

Windowing tools are best suited for lower end systems where memory is constrained and the interface must be tightly integrated with the system. The windowing tools examined in connection with this thesis were mainly PC versions that allowed pop-up, pull-down, mouse-driven menu systems. They were written mostly in C, as C is not only very popular for this purpose, but also integrates very well with CLIPS. These systems were mostly shareware products, so the cost is very low.

This type of interface requires more programmer involvement to implement as the manipulation of the code to modify for system-specific use is at a lower level of abstraction than the more object oriented interfaces. Though these tools are easily modifiable by a programmer, the instructor would need to have an understanding of the programming language, which is not desirable for this shell. The programmer may be able to modify these tools, given the source code is available with each product.

The main benefit of these tools is that once integrated, the system is more tightly integrated and specialized for faster, more efficient operation. The closer the link between the components, the less time required to pass information between them for execution. This is an important consideration when trying to minimize the student-machine interaction time. Graphics and data passing can require a great deal of time to complete while the student waits. Again, this is dependent on the implementation platform.

2. Toolbook, by Asymetrix

Toolbook is a software construction tool that allows the user to use a graphical user interface and object oriented programming features to develop applications tailored for their specific purpose. Toolbook is a hypertext system that only runs under Microsoft's Windows, on the PC. It allows the user to create screens, called pages, that are linked and designed to carryout actions through scripts (small programs attached to objects that cause an action to occur). (Asymetrix, 1989, pp. 2-16)

Figures 9a through 9h give an example of the capabilities of Toolbook. They are taken from the systems "Quick Tour" book which was developed using Toolbook and comes with the system. These "pages" provide highlights of some of the capabilities of Toolbook. These figures provide good examples of the application's ability to perform at the student, or reader, level and at the instructor, or author, level.

Toolbook is currently being used at the Defense Language Institute (DLI), at the Presidio in Monterey, California, as the interface to the Microsoft Windows compatible version of Computer Assisted Language Instruction System (WinCALIS), a

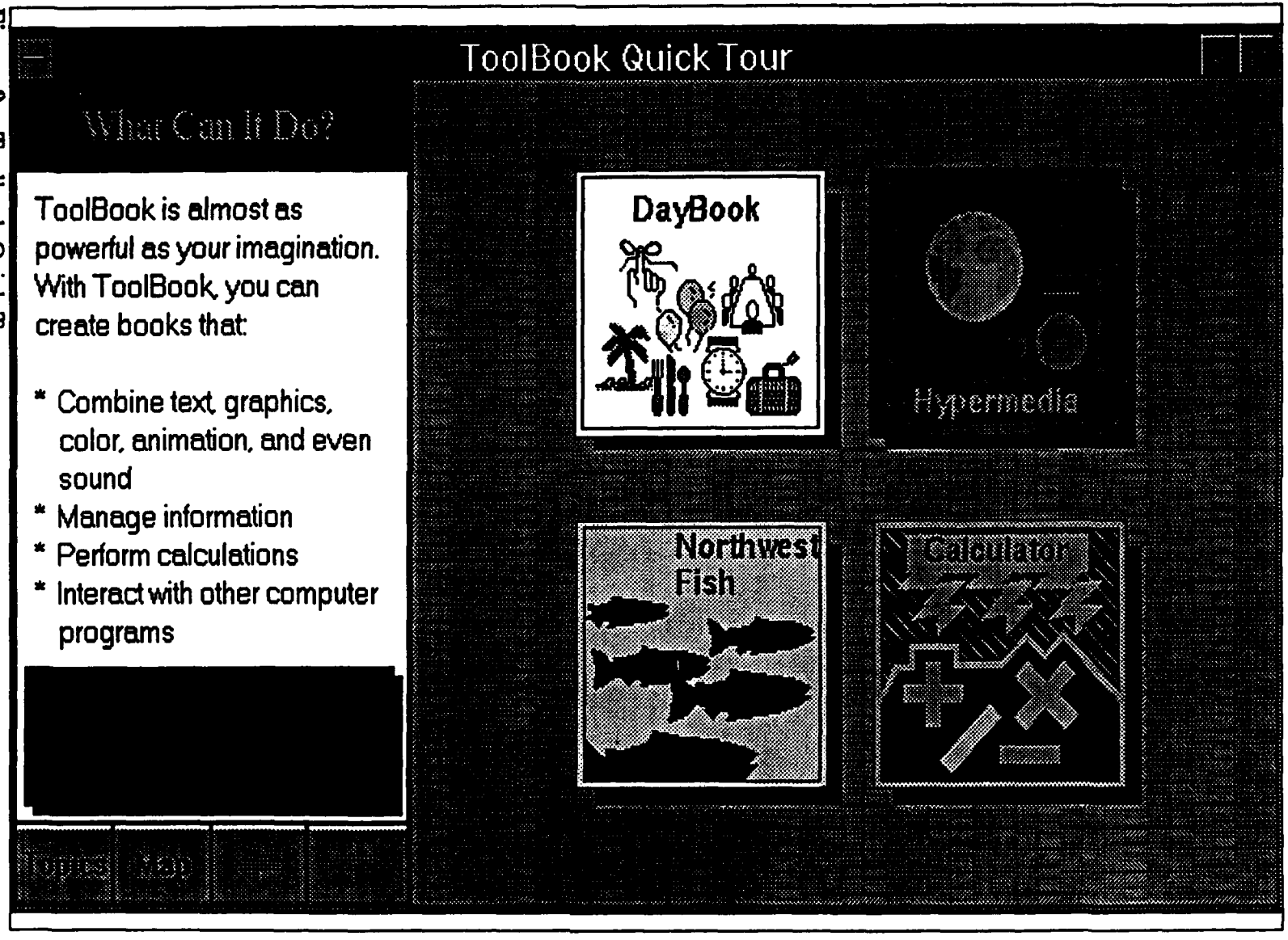


Figure 9a Toolbook Quick Tour

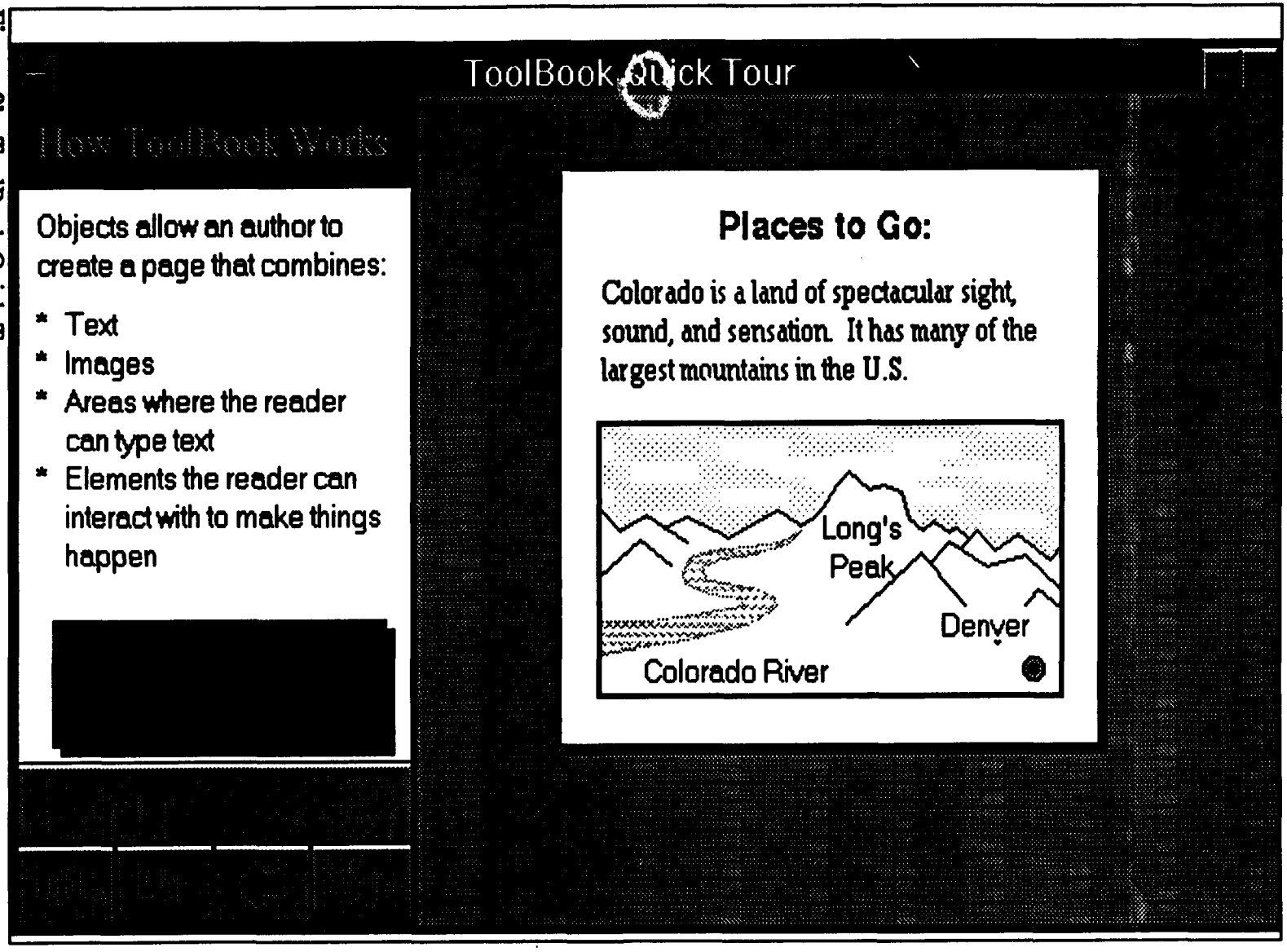


Figure 9b ToolBook Quick Tour

Figure 9c ToolBook Quick Tour

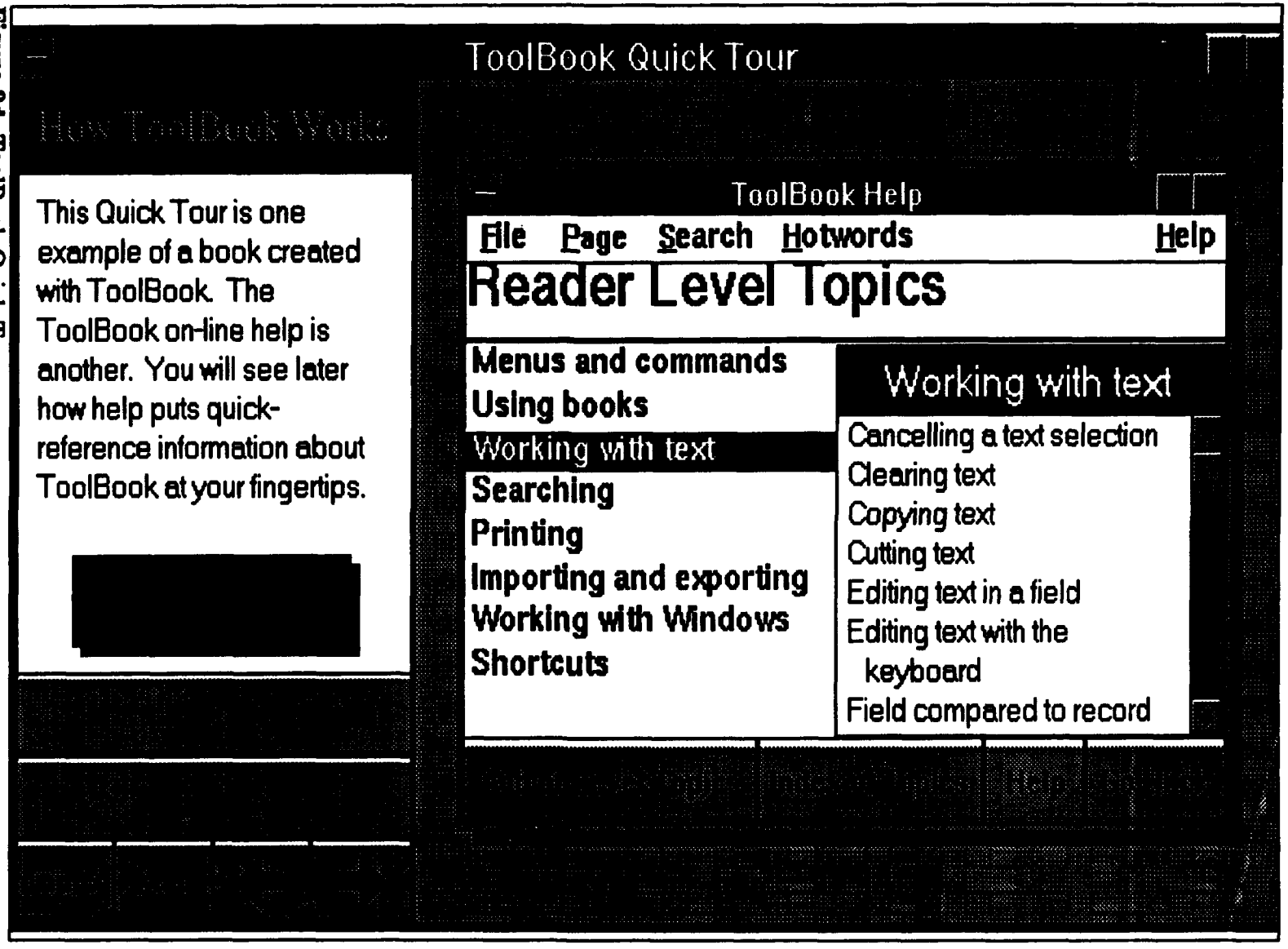


Figure 9d ToolBook Quick Tour

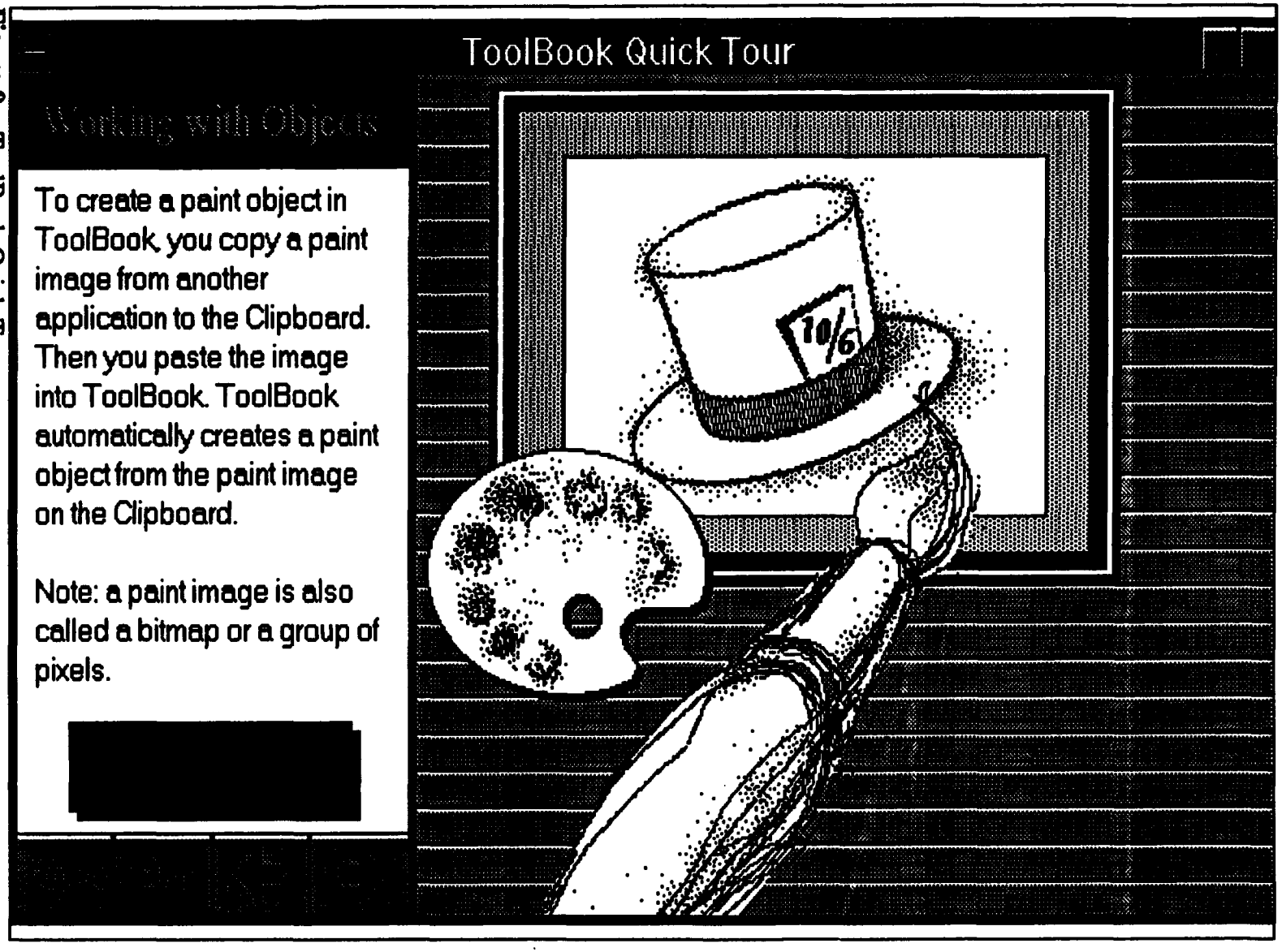


Figure 9c ToolBook Quick Tour

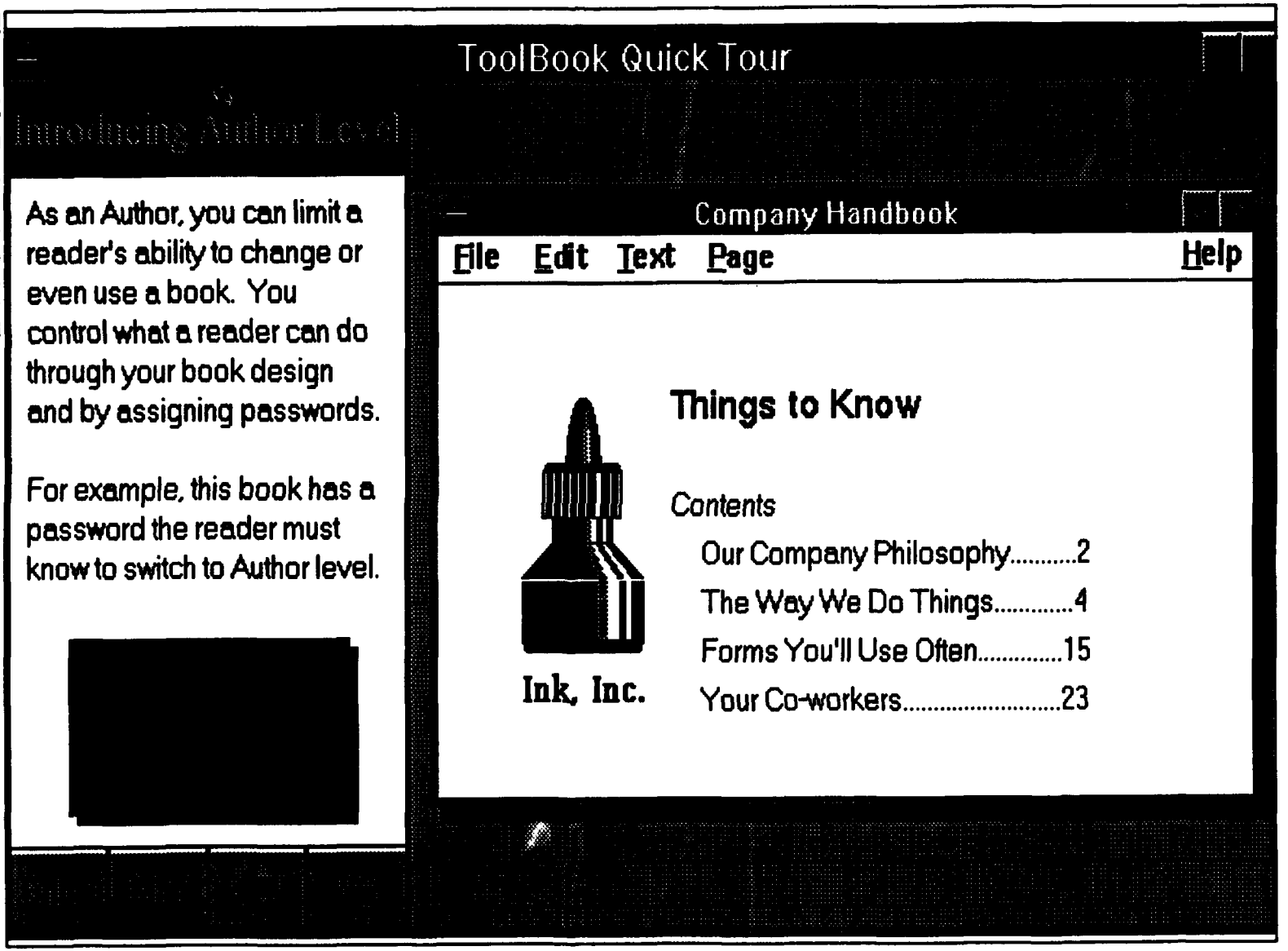


Figure 9f ToolBook Quick Tour

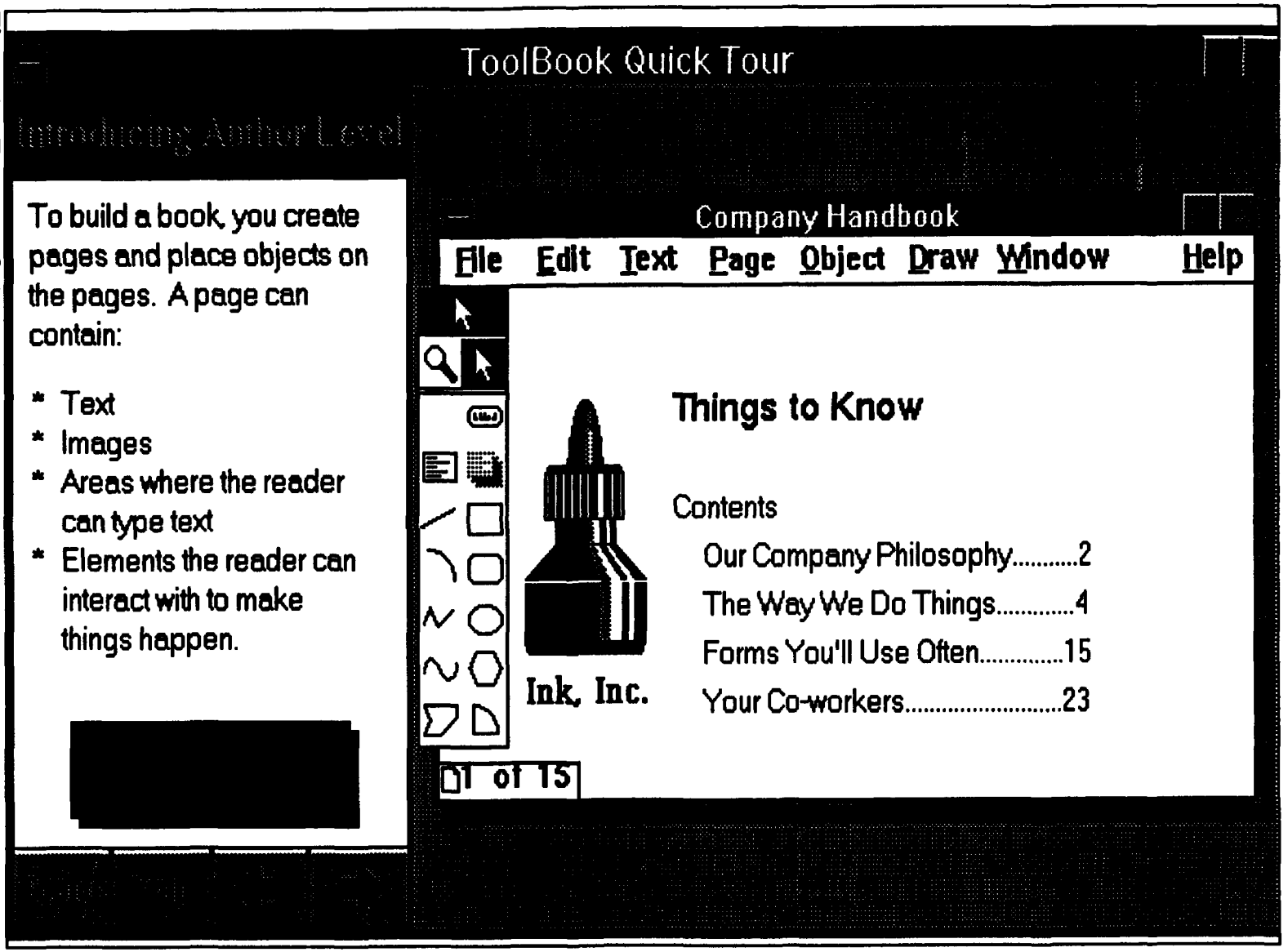


Figure 9g ToolBook Quick Tour

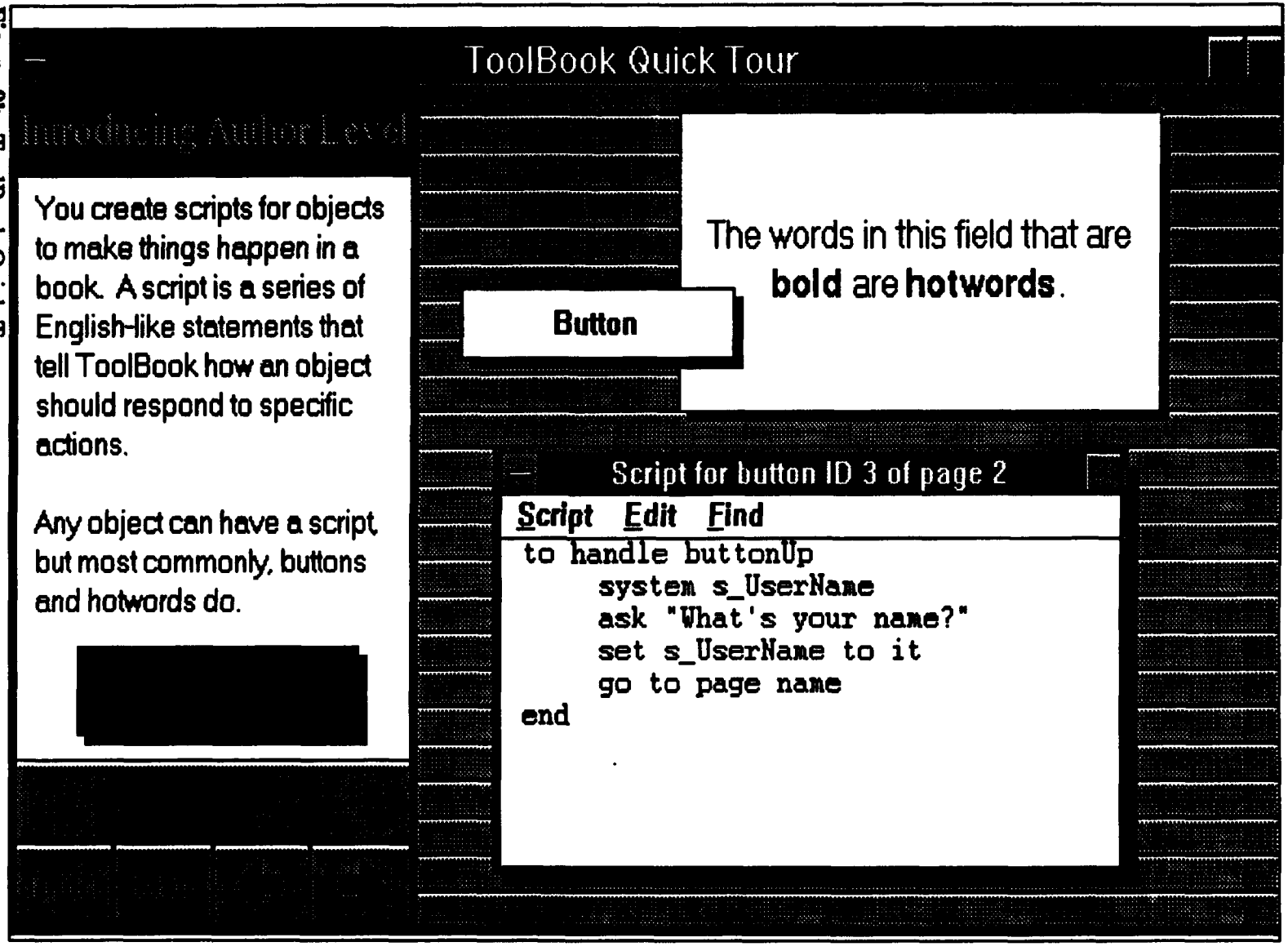


Figure 9h ToolBook Quick Tour

CAI tool developed by Duke University. This combination, coupled with video disk technology, allows instructors to create multimedia presentations of situations in various cultures to instruct students in situational conversations in the language being taught. This system could be enhanced by using CLIPS to incorporate student modeling and instructional strategy adaptation provided by an ITS. The medium is in place to allow the instructor to integrate the interface with the presentation of instructional material, but there is no intelligent component incorporated into the system.

3. ObjectVision as an Interface

ObjectVision for Windows was created by Borland International as an interactive tool to allow nonprogrammers the ability to create custom-made applications for Microsoft's Windows. ObjectVision is a visual programming tool that allows for a layered construction which can incorporate the use of different applications through dynamic links and working with objects and forms. It uses decision trees to control the logic of the application and the assignment of values to field objects. This tool was designed for the incremental development of applications and is intended to be easily modified by the developer. Figures 10a and 10b show the interactive methods the instructor would use to establish links and create his interface. (Borland, 1991, pp. 7-19)

C. THE EXPERT MODULE AND DOMAIN KNOWLEDGE

The ITS will only be as useful and reliable as the knowledge it is trying to impart to the student. The methods of presentation are an integral part of determining how the knowledge is stored. There are many tools available commercially, for many different

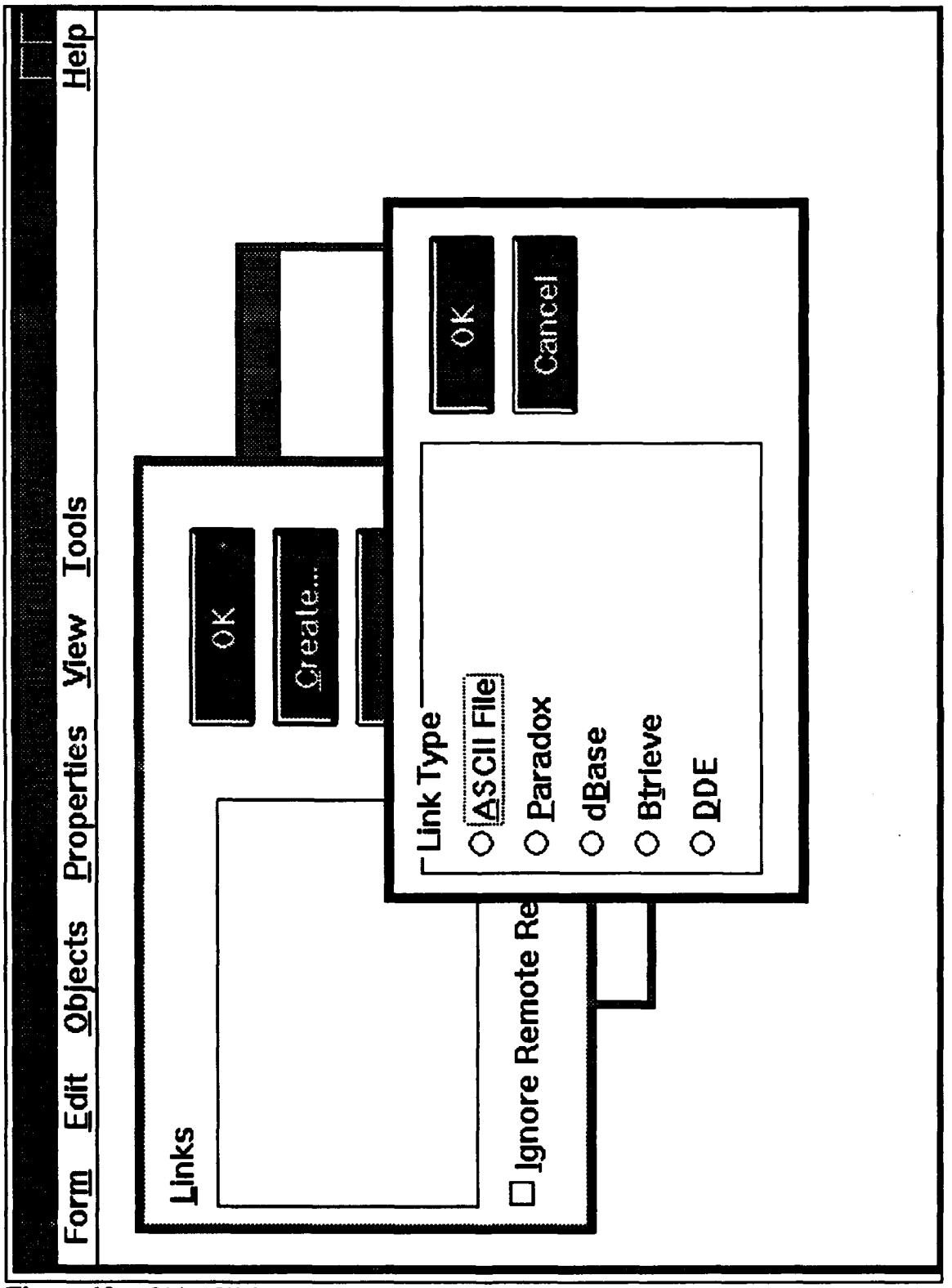


Figure 10a ObjectVision

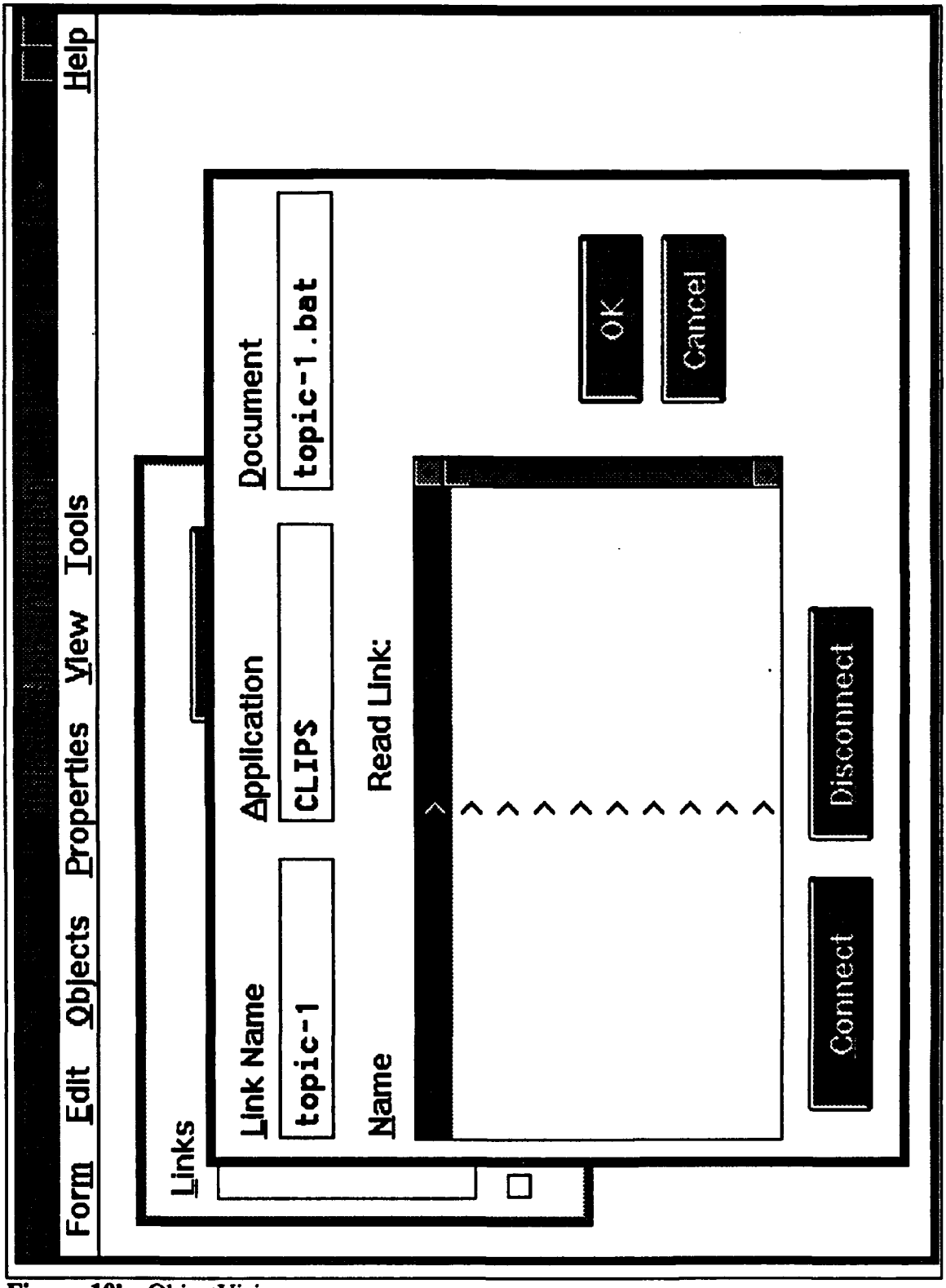


Figure 10b ObjectVision

computer platforms, that can assist the educator in getting the material across to the student. Although other platforms will be discussed, the experimentation was conducted on the PC.

Knowledge bases constructed to form the expert module can be difficult and time consuming to develop. By using the OOP approach, the instantiations of domain independent objects could be created for domain specific knowledge. Through data encapsulation and the inheritance of methods, or message handlers, and characteristics, the knowledge bases could be fit to developed ITSs built using similar structures. For example, a domain independent method sent to an object instructing the object to show a presentation, could be made domain dependent by the presentation of a specific type developed specifically for that object.

To demonstrate this situation more clearly, a test was conducted using different presentation tools and calling the presentation from a CLIPS rule. Once the student's cognitive state has been determined, the student model has been examined, and the student has requested information on a certain topic, a CLIPS rule would fire. The resulting action would be the presentation of the appropriate material for that student at the time determined by the system. The example rule would look similar to the following example:

```
(defrule SHOW-TOPIC
   (need-info-about ?topic)
=>
   (bind ?presentation (str-cat rundemo ?topic))
   (system ?presentation))
```

This rule firing would cause a presentation of the topic designated and developed by the instructor to be shown to the student. The system could then proceed to the next determined stage of test, evaluation, a more detailed presentation, or whatever was appropriate for the individual student.

1. Presentation Tools

Many commercially available presentation tools can be used to assist the instructor in presenting the required knowledge. Some of these tools are more interactive than others. They also have different requirements for creating stand-alone, modular presentations. For example, CLIPS was used with Genus Microprogramming's Proteus to create individual lessons for the two types of aircraft from ART (Campbell, 1990). This experiment was conducted from the DOS environment, using Microsoft's DOS 5.0, and from within Microsoft's Windows 3.0. WordPerfect Corporation's DrawPerfect presentation feature was also called from CLIPS in these two different environments to show the feasibility of using different presentation tools with CLIPS.

A beta release of Microsoft's new Multimedia Development Kit (MDK) was also examined at DLI for this thesis as a more complex tool available for the presentation of material. As this tool is still in the development stage, there are still programming flaws being corrected. As these "bugs" are fixed, a future goal is to experiment with integrating CLIPS with presentation sequences to add intelligence to this powerful new presentation tool. As the system is currently configured, there is virtually little "intelligence" incorporated into the system.

The MDK is linked with Toolbook and WinCALIS to form the Linguist Workstation. Figures 11a to 11h give examples of the capabilities the MDK. Figure 11a shows the different languages available to teach with this system. The example language chosen in Figures 11b through 11d is German. Figures 11b and 11c show the student level of interaction available. Figure 11d shows the instructor, or authoring, level of interaction for the same "page" of this lesson. The instructor can switch between levels. The student, however, is restricted to only the student level.

The Linguist Workstation demonstrates the layered approach proposed for the ITS shell in this thesis. This system is currently capable of running only on a PC, and requires addition hardware, such as a video disk and a sound board. Therefore, this system has very limited component portability.

D. THE STUDENT DIAGNOSTIC MODULE

The student diagnostic module should give the current state of the student in his understanding of the knowledge domain being taught. As stated earlier, this can be done by recording facts about the student's cognitive process. As the facts in the system change, the system should adjust the instructional material being presented, the teaching strategy being used, or the evaluation and critiquing process as needed to best fit the needs for the individual interacting with the system.

Determining the student's initial state can be done in a number of different ways. The system can make an assumption and start all students at the same point and adjust the student model as the student demonstrates his level of expertise. This seems to be

Figure 11a Multimedia Development Kit Linguist Workstation Editor

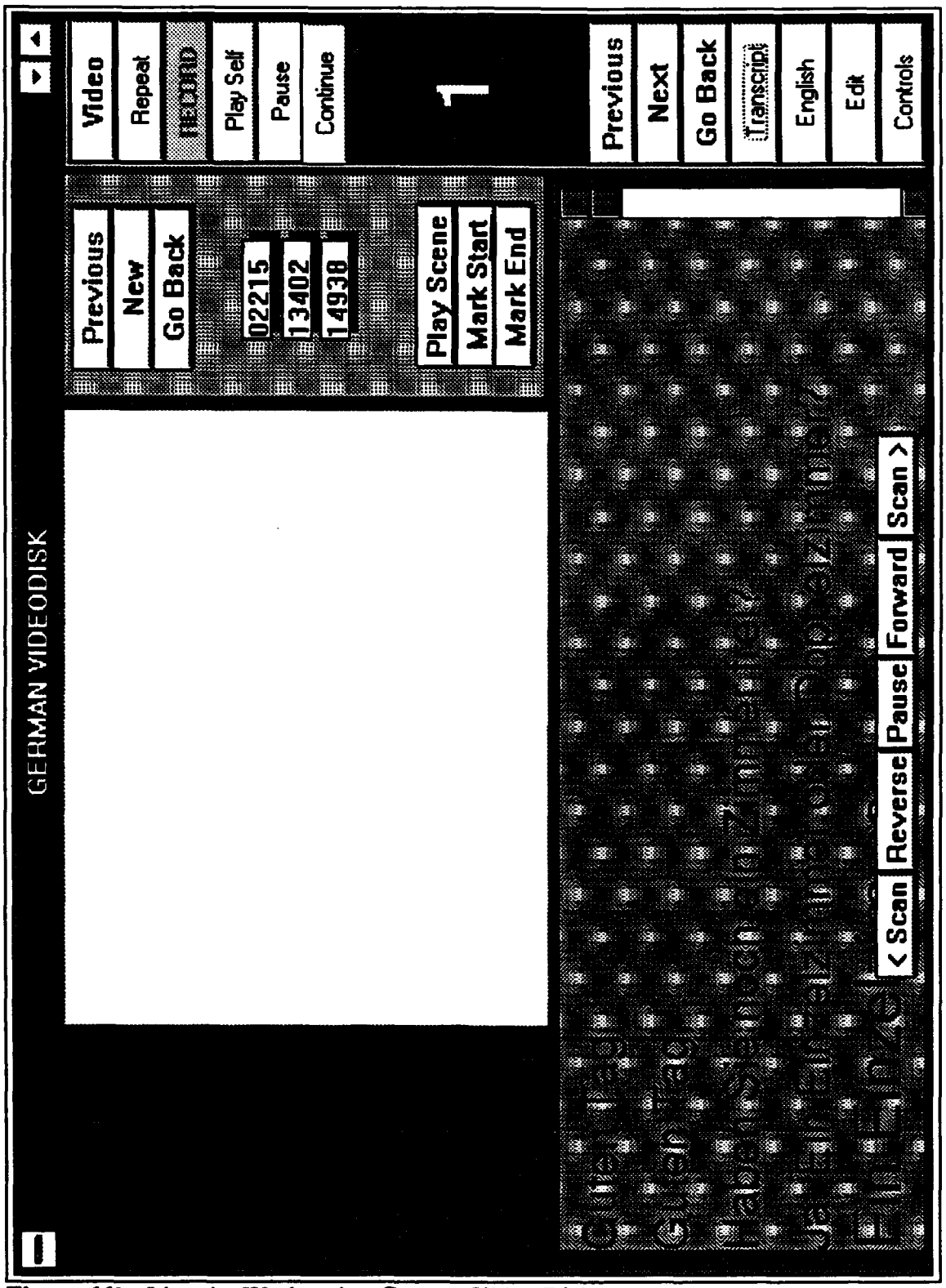


Figure 11b Linguist Workstation German Video Disk

Figure 11c Linguist Workstation German Video Disk in English

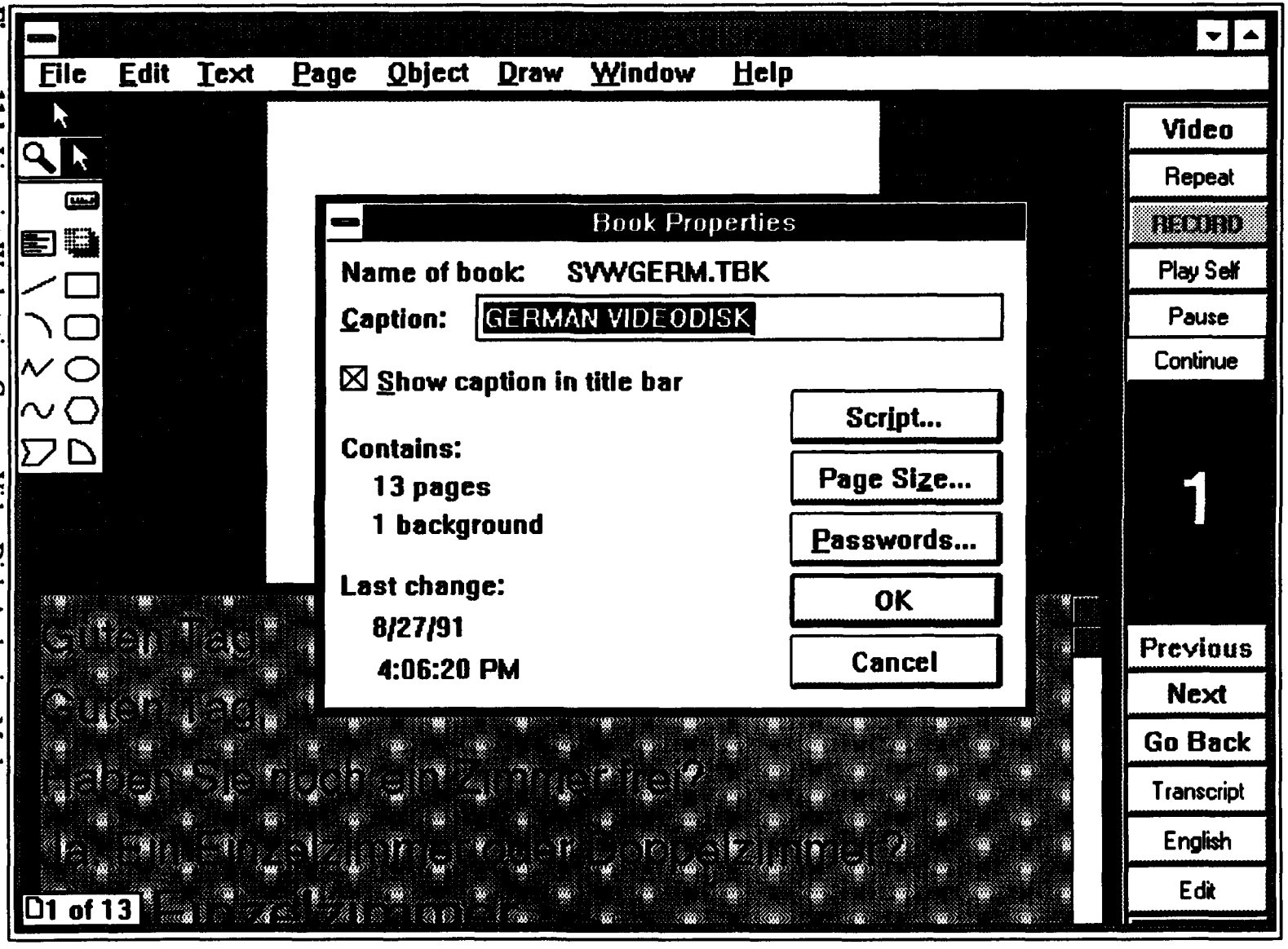


Figure 11d Linguist Workstation German Video Disk Authoring Mode

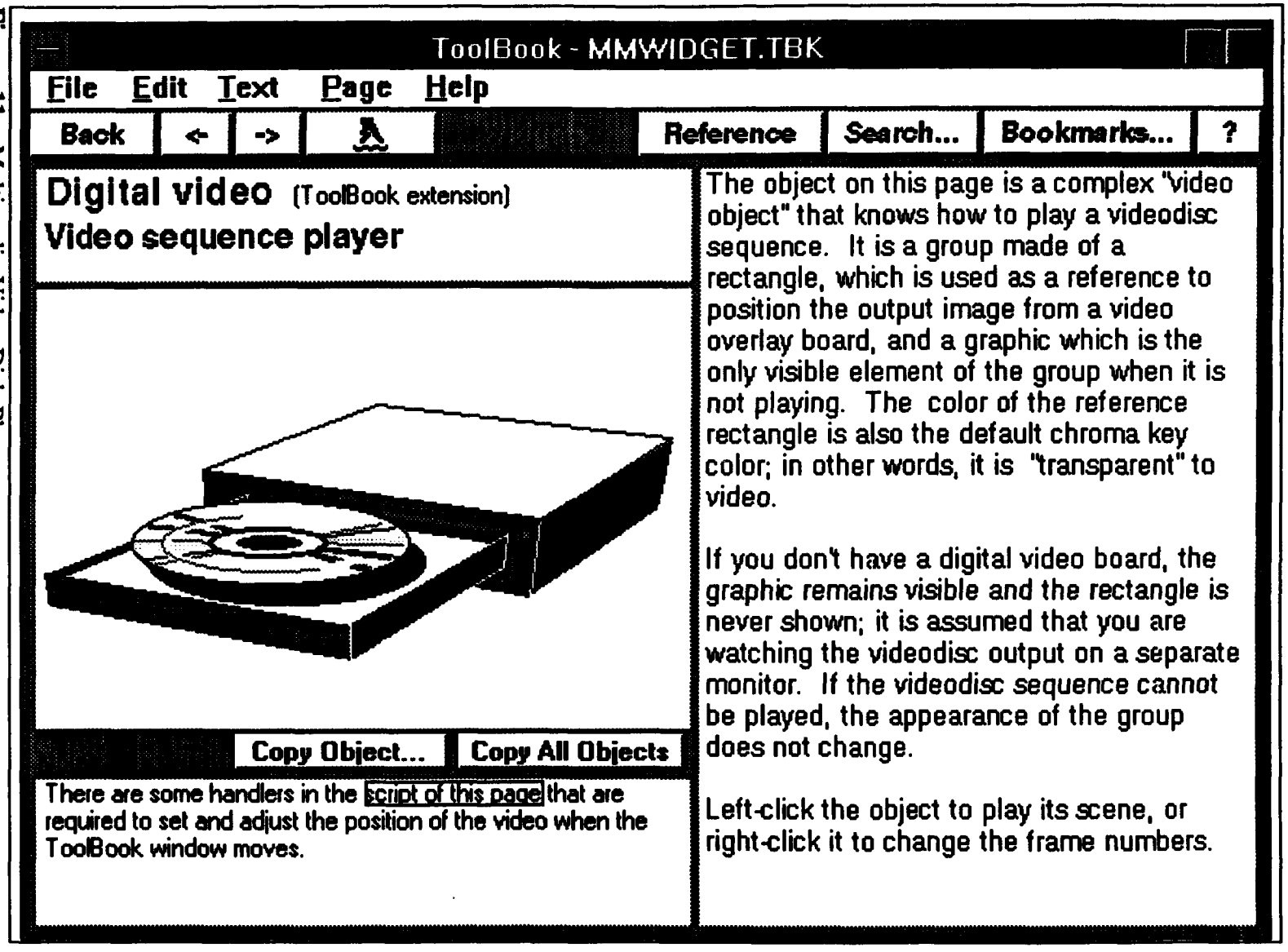


Figure 11e Multimedia Video Disk Player

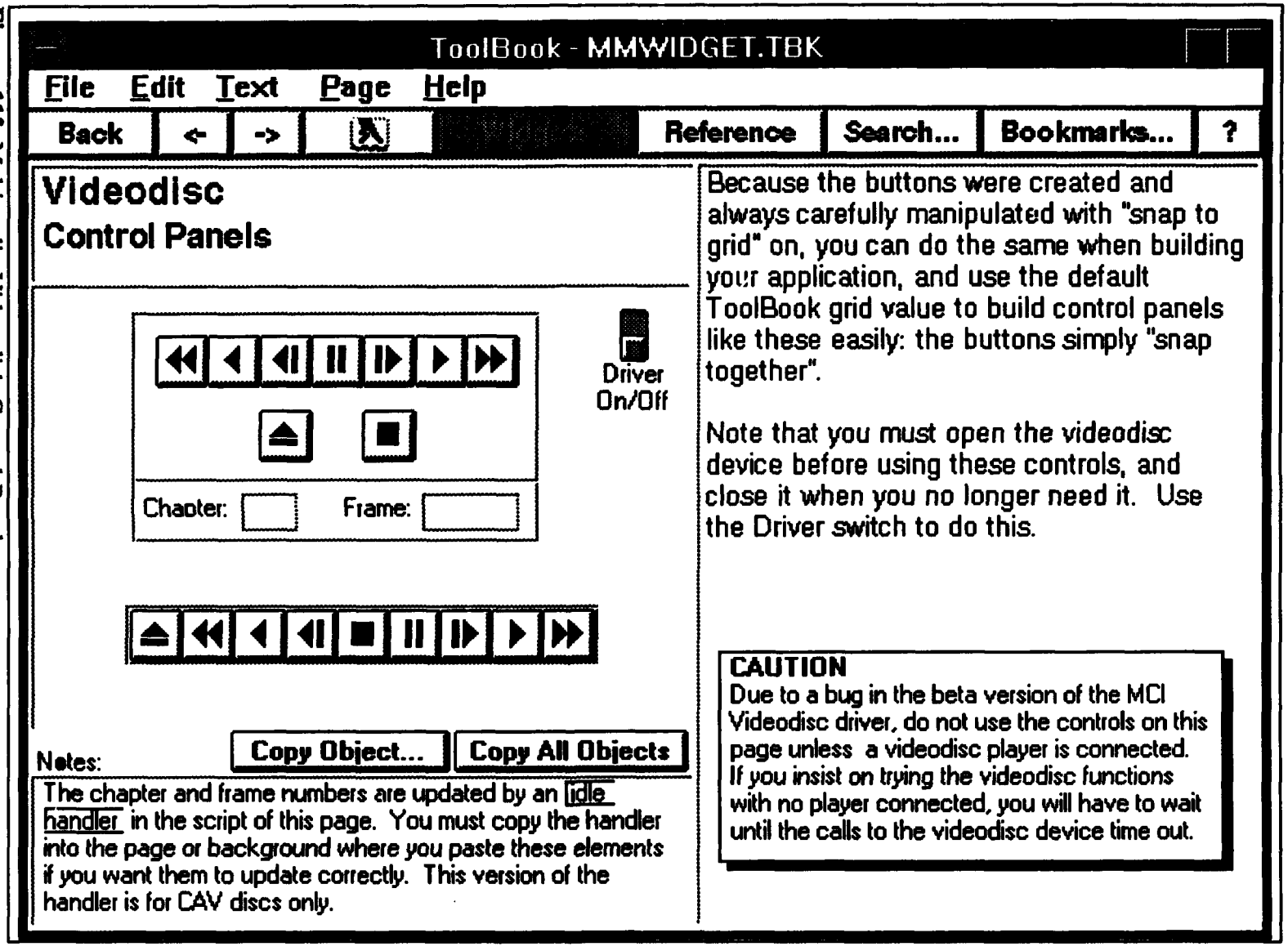


Figure 11f Multimedia Video disk Control Panel

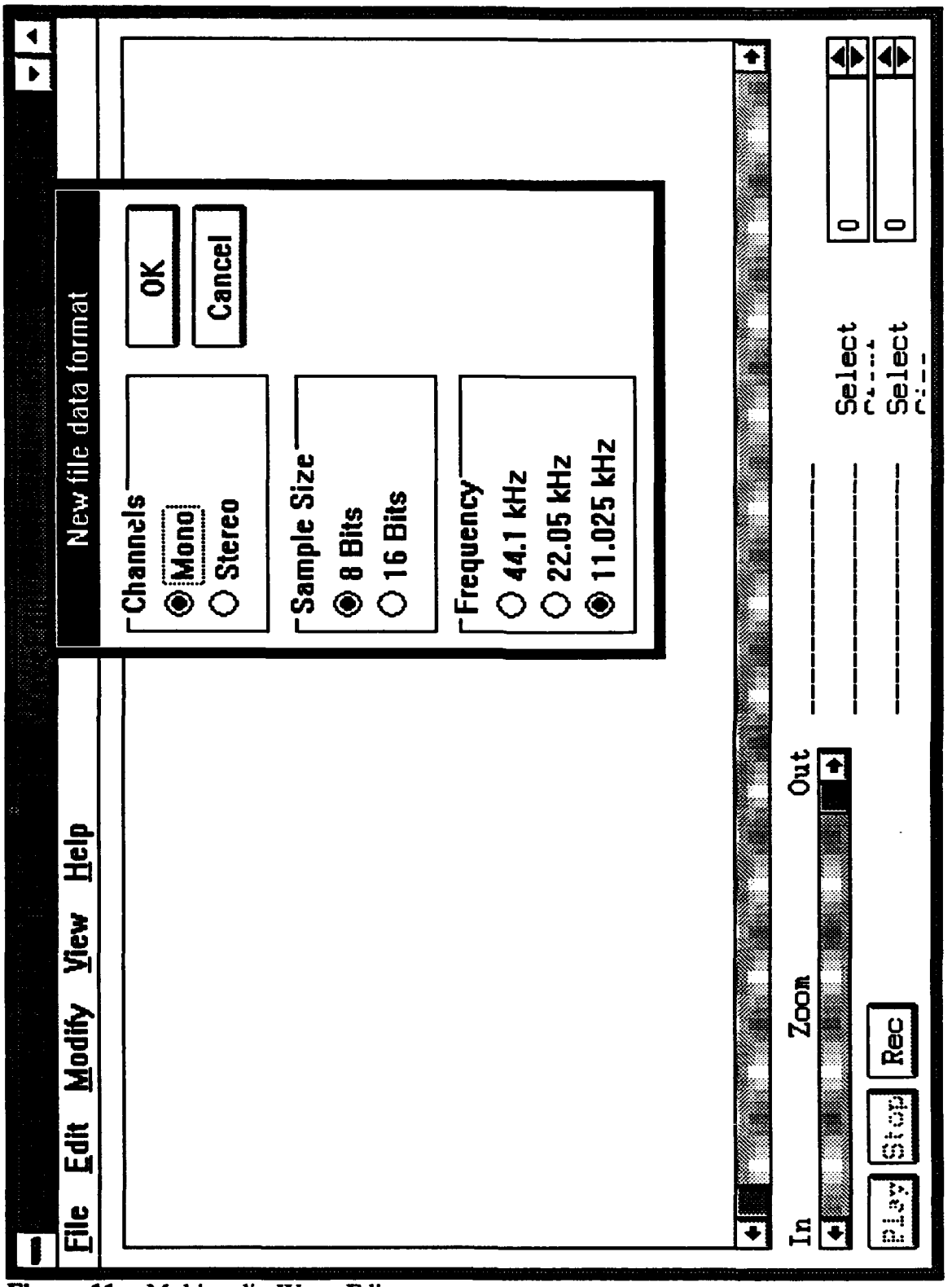


Figure 11g Multimedia Wave Editor

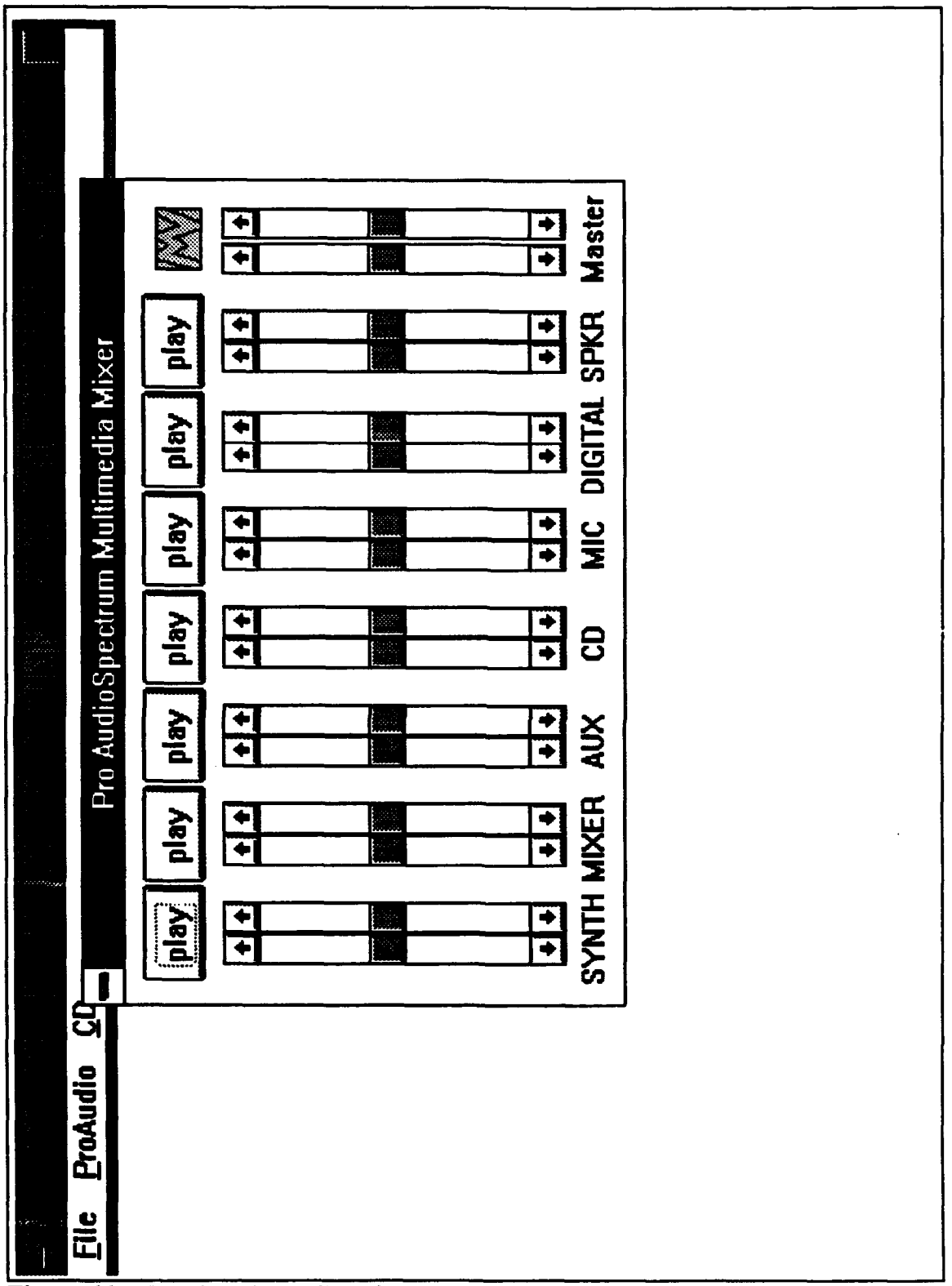


Figure 11h Multimedia Audio Mixer

the easiest way for the programmer, but does not give the student the proper starting point for his level of knowledge of the subject.

The system could also begin by conducting a survey by requiring the student to answer some initial questions to determine his background before beginning the instruction. This can be frustrating for the student who is more concerned with getting into the material than answering a survey. One technique that is widely used in the military is conducting a pretest to determine the student's initial understanding of the topic, train the student based on the pretest, conduct an evaluation after the instruction, and then re-train based on the evaluation. These are just a few techniques. The possible options and applicable methods should be developed by the cognitive scientists and behavioral specialists. The options should be selectable by the instructor and he should be prompted for the required information to instantiate that option.

E. THE INSTRUCTIONAL MODULE

The instructional module is an integral part of this system, also. The same time of options should be provided as in the student diagnostic module. There are multiple strategies that can be invoked at different times as the student progresses through the desired lessons. A major consideration in using these strategies is to allow the student to remain in control of the interaction with the system. These strategies are there to provide the student with the most efficient means to learn the required material.

This module needs considerable more effort and examination than was provided in this thesis. The tools examined were geared toward a high level of student control and

interaction. The proper mix of integration of this module with the remainder of the system must still be examined. The tools are there, however, the ability to codify these strategies must come from the cognitive scientists or behavioral psychologists. Once programmed, these components should be place into a selectable option in a form that the instructor can instantiate for his system.

F. SUMMARY

There are many powerful applications available to perform the required functions of the modules in an ITS. The ITS shell proposed in this thesis stresses the layered approach to implementing this system. Allow the user to achieve his goals in using this system without influencing the levels below him. The goal is to provide the instructor with a workable product to produce an ITS in the most efficient and reliable means possible. The system must stress modularity, portability, and abstraction. The area experts should produce the components, or subcomponents required for each module. The system should also provide for incremental development.

Numerous systems and tools were examined to show the feasibility of linking these programs into the ITS structure outlined in this thesis. Although many of the more complex systems are new and have gone through limited testing and evaluation in ITS applications, the experimental results are encouraging. The tools are available to help ITSs make the transition from the research arena into the instructional world. It is time to put forth a joint effort, by all research area specialist involved in the development an ITS, and begin producing such products for the educator.

VI. CONCLUSIONS

A. LESSONS LEARNED

ITSs can cover a wide spectrum of applications, knowledge domains and levels of complexity. The expectation of being able to use a generic shell to develop an ITS that covers this broad spectrum is unrealistic. It is possible, however to develop domain independent components and use domain or system dependent instantiations of these components to build more simplistic ITSs.

The goal is to provide instructors with a tool that would assist them in tailoring the components of an ITS to fit their instructional needs. By allowing the instructor to select those components that he can use and get a working system is a significant achievement. If the emphasis of modular design and portability are adhered to, the system can continually be improved without discarding all previous work. Reuse of code and various components can possibly allow more systems to become available for use and evaluation. As the different research areas begin to work closer together and products produced by area experts are better integrated, systems will increase in complexity and reliability.

Building an ITS with current technology, on lower end platforms, highlights the tradeoffs needed to balance components to fit within system constraints. Current tools that incorporate extensive graphics, such as Toolbook are extremely slow in execution. Other programs that allow passage of data and command to other applications also required a great deal of processing time. The memory constraints that these tools must

operate in are also a problem with current technology. For example, CLIPS must be stripped down to just essential components to allow integration with more complex applications that required large amounts of memory.

With the pace of technological advances in memory utilization and system speeds, it is evident that these constraints will not be long lasting. The fact that it is possible to link the various applications is proof enough that modular design and integration of the components proposed is possible. A major concern with the use of any system is the amount of time and effort that must be expending learning the new system. A training program with hands-on examples must be an integral part of using a system as described in Chapter V. The goal is to minimize the amount of extra training, but to give the instructor and the student confidence in using this shell, training should be a major area of emphasis.

B. ACCOMPLISHMENTS

CLIPS was thoroughly examined and modified to provide an effective vehicle to control the components of an ITS. By recompiling the CLIPS source code, CLIPS was tested to function as both a developmental tool and in an implementation configuration to maximize the amount of memory for external calls to other programs from inside CLIPS. After developing modular presentations of different topics, CLIPS was able to invoke the presentations. This test was conducted in the MS-DOS environment and running under Microsoft Windows. This functionality demonstrates the ability of CLIPS

to control the presentation of material by requiring a certain state, facts in the fact base, to cause a rule to fire and to present the needed material as it's action.

Also, by experimenting with various interface tools and presentation tools, it was shown that the incorporation these existing tools as components of the modules required for ITS development is possible. Since many of these tools are very new, it was not possible to conduct extensive evaluation to the possible problems that may arise, but successful integration was achieved. The ease of use that the developers of these tools propose appears to be overstated, however. After examination for ease of integration, the claims that nonprogrammers could easily create there own custom-made applications seems quite optimistic.

C. FUTURE WORKS AND MODIFICATIONS

To fully test the structure of the ITS shell, different components for the shell need to be developed. These different components should incorporate the current theories of instructional techniques. They should also be written in CLIPS for maximum portability. Once programmed, they should be configured to enable an instructor to select the appropriate component, or subcomponent, to suit his given application.

Once a full complement of components are available, a completed system could be produced, tested, and evaluated. The components should be tested for ease of integration with existing interface and presentation tools. This would allow the instructor to develop an ITS by concentrating more on the interaction with the student and the material to be presented.

As systems are implemented, the ease of transporting the components to other platforms could be examined. The transporting of these components should as transparent to the user and instructor, as possible, to reduce the impact of a new learning curve on the new system.

LIST OF REFERENCES

Anderson, John R. "The Expert Module." In Foundations of Intelligent Tutoring Systems. Ed Martha C. Polson and J. Jeffrey Richardson. New York: Lawrence Erlbaum Associates, 1988, pp 21-53.

Asymetrix Corporation. Using Toolbook: A Guide to Building and Working with Books. Asymetrix Corporation, 1989.

Baudendistel, Stephen. "Consider CLIPS...." In AI Exchange. Spring, 1990, pp. 3-14.

Bielawski, Larry and Lewand, Robert. Intelligent Systems Design: Integrating Expert Systems, Hypermedia, and Database Technologies. New York: John Wiley & Sons, 1991.

Bonar, Jeffrey G. "Interface Architecture for Intelligent Tutoring Systems." In Intelligent Tutoring Systems: Evolutions in Design. Ed. Hugh Burns, James W. Parlett, and Carol Luckhardt Redfield. New Jersey: Lawrence Erlbaum Associates, 1991, pp 35-67.

Bower, M. and Hilgard, J. Theories of Learning. New York: Addison-Wesley Publishing Company, 1981.

Burns, Hugh L. and Capps Charles G. "Foundations of Intelligent Tutoring Systems: An Introduction." In Foundations of Intelligent Tutoring Systems. Ed Martha C. Polson and J. Jeffrey Richardson. New York: Lawrence Erlbaum Associates, 1988, pp 1-19.

Burns, Hugh and Parlett, James W. "The Evolution of Intelligent Tutoring Systems: Dimensions of Design." In Intelligent Tutoring Systems: Evolutions in Design. Ed. Hugh Burns, James W. Parlett, and Carol Luckhardt Redfield. New Jersey: Lawrence Erlbaum Associates, 1991, pp 1-11.

Campbell, Larry W. "An Intelligent Tutor System for Visual Aircraft Recognition." Master's Thesis, Naval Postgraduate School, Monterey, California, June 1990.

Citrenbaum, Ronald, Geissman, James R., and Schultz, Roger. "Selecting A Shell." AI Expert, September 1987, pp 30-39.

CLIPS Reference Manual, Volume I, Basic Programming Guide. Software Technology Branch, NASA-Lyndon B. Johnson Space Center, 1991.

Donnell, Brian. Electronic Mail responses concerning CLIPS, NASA's CLIPS Development Team, 19 August 1991.

Elliot, David B. "An Introduction to Object-Oriented Programming." In AI Exchange. Spring, 1990, pp. 20-22.

Engelmore, Robert, Morgan, Tony, and Nii, H. P. "Introduction," In Blackboard Systems. Ed. Robert Engelmore and Tony Morgan. New York: Addison-Wesley Publishing Company, 1981, pp. 1-22.

Giarratano, Joseph C. CLIPS User's Guide: Volume I, Rules, CLIPS Version 5.0, NASA, Johnson B. Space Center, Artificial Intelligence Section, January 1991.

Giarratano, Joseph C. CLIPS User's Guide: Volume II, Objects, CLIPS Version 5.0, NASA, Johnson B. Space Center, Artificial Intelligence Section, May 1991.

Hill, Randall W. Jr. and Pickering, Brad. "Intelligent Tutoring Using HyperCLIPS." First CLIPS Conference Proceedings. NASA-Johnson Space Center, August 1990, pp 62-68.

Hua, Grace. "Developing an Intelligent Computer-Aided Trainer." First CLIPS Conference Proceedings. NASA-Johnson Space Center, August 1990, pp 69-74.

Inui, Masahiro, Miyasaha, Nobuji, Kawamura, Kozuhika, and Bourne, John R. "Development of a Model-Based Intelligent Training System." North-Holland, Future Generation Computer Systems 5, 1989, pp 59-69.

Khoshafian, Setrag and Abnous, Razmik. Object Orientation: Concepts, Languages, Databases, User Interfaces. New York: John Wiley & Sons, 1990.

Lenat, Douglas B. and others. "Cyc: Toward Programs with Common Sense." Communications of the ACM. v. 33, August 1990, pp. 30-49.

Murray, Tom and Woolf, Beverly. "A Knowledge Acquisition Tool for Intelligent Computer Tutors." In Sigart Bulletin, April 1991, pp 9-21.

Office of Artificial Intelligence Analysis and Evaluation. "Intelligent Computer-Aided Instructional Systems." In AI Exchange. January-March, 1989, pp. 6-7.

O'Neil, Harold F. Jr., Slawson, Dean A., and Baker, Eva L. "Design of a Domain-Independent Problem-Solving Instructional Strategy for Intelligent Computer-Assisted Instruction." In Intelligent Tutoring Systems: Evolutions in Design. Ed. Hugh Burns, James W. Parlett, and Carol Luckhardt Redfield. New Jersey: Lawrence Erlbaum Associates, 1991, pp 69-103.

Pirolli, Peter. "Computer-Aided Instructional Design Systems." In Intelligent Tutoring Systems: Evolutions in Design. Ed. Hugh Burns, James W. Parlett, and Carol Luckhardt Redfield. New Jersey: Lawrence Erlbaum Associates, 1991, pp 105-125.

Redfield, Carol Luckhardt and Steuck, Kurt. "The Future of Intelligent Tutoring Systems." In Intelligent Tutoring Systems: Evolutions in Design. Ed. Hugh Burns, James W. Parlett, and Carol Luckhardt Redfield. New Jersey: Lawrence Erlbaum Associates, 1991, pp 265-284.

VanLehn, Kurt. "Student Modeling." In Foundations of Intelligent Tutoring Systems. Ed Martha C. Polson and J. Jeffrey Richardson. New York: Lawrence Erlbaum Associates, 1988, pp 55-78.

Woolf, Beverly. "Intelligent Tutoring Systems, A Survey." Morgan Kaufmann Publishers, 1988, pp 1-44.

Woolf, Beverly. "Representing, Acquiring, and Reasoning About Tutoring Knowledge." In Intelligent Tutoring Systems: Evolutions in Design. Ed. Hugh Burns, James W. Parlett, and Carol Luckhardt Redfield. New Jersey: Lawrence Erlbaum Associates, 1991, pp 127-149.

Woolf, Beverly Park, et al. "Knowledge-based Environments for Teaching and Learning." AI Magazine, Special Issue, 1991, pp 74-77.

BIBLIOGRAPHY

Booch, Grady. Object Oriented Design with Applications. Redwood City: Benjamin/Cummings Publishing Company, 1991.

Boy, Guy A. Intelligent Assistant Systems. San Diego: Academic Press, 1991.

Burton, Richard R. "The Environment Module of Intelligent Tutoring Systems." In Foundations of Intelligent Tutoring Systems. Ed Martha C. Polson and J. Jeffrey Richardson. New York: Lawrence Erlbaum Associates, 1988, pp 109-142.

Fink, Pamela K. "The Role of Domain Knowledge in the Design of an Intelligent Tutoring System." In Intelligent Tutoring Systems: Evolutions in Design. Ed. Hugh Burns, James W. Parlett, and Carol Luckhardt Redfield. New Jersey: Lawrence Erlbaum Associates, 1991, pp 195-224.

Giarratano, Joseph and Riley, Gary. Expert Systems: Principles and Programming. Boston: PWS-Kent Publishing, 1989.

Halff, Henry M. "Curriculum and Instruction in Automated Tutors." In Foundations of Intelligent Tutoring Systems. Ed Martha C. Polson and J. Jeffrey Richardson. New York: Lawrence Erlbaum Associates, 1988, pp 79-108.

Merrill, David M. "An Expert System for Instructional Design." IEEE Expert, Summer 1987, pp 25-37.

Mettrey, William. "A Comparative Evaluation of Expert System Tools." In Computer, February 1991, pp 19-31.

Miller, James R. "The Role of Human-Computer Interaction in Intelligent Tutoring Systems." In Foundations of Intelligent Tutoring Systems. Ed Martha C. Polson and J. Jeffrey Richardson. New York: Lawrence Erlbaum Associates, 1988, pp 143-189.

Mueller, Stephen J. "Incorporating CLIPS into a Personal-Computer-Based Intelligent Tutoring System." First CLIPS Conference Proceedings. NASA-Johnson Space Center, August 1990, pp 75-79.

Nicol, Anne. "Interfaces for Learning: What Do Good Teachers Know That We Don't?" In The Art Of Human-Computer Interface Design. Ed. Brenda Laurel. Reading, Massachusetts: Addison-Wesley Publishing Company, 1991.

Ragsdale, Daniel J. and Tidd, John P. "Designing Intelligent Computer Aided Instruction Systems with Integrated Knowledge Representation Schemes." Master's Thesis, Naval Postgraduate School, Monterey, California, June 1990.

Richer, Mark H. "An Evaluation of Expert System Development Tools." In AI Tools and Techniques. Ed. Mark H. Richer. New Jersey: Ablex Publishing, 1989, pp 67-105.

Swigger, Kathleen M. "Managing Communication Knowledge." In Intelligent Tutoring Systems: Evolutions in Design. Ed. Hugh Burns, James W. Parlett, and Carol Luckhardt Redfield. New Jersey: Lawrence Erlbaum Associates, 1991, pp 13-34.

Tailor, Anita. "MXA -- A Blackboard Expert System Shell." In Blackboard Systems. Ed Robert Engelmore and Tony Morgan. New York: Addison-Wesley Publishing Company 1988, pp 315-333.

Zanconato, Roberto. "BLOBS -- An Object-Oriented Blackboard System Framework for Reasoning in Time." In Blackboard Systems. Ed Robert Engelmore and Tony Morgan. New York: Addison-Wesley Publishing Company 1988, pp 335-345.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
4.	Dr. Robert B. McGhee CS/Mz Naval Postgraduate School Monterey, California 93943-5000	2
5.	Dr. Yuh-jeng Lee CS/Le Naval Postgraduate School Monterey, California 93943-5000	6
6.	David Pratt Naval Postgraduate School Code CS, Department of Computer Science Monterey, California 93943-5000	1
7.	CPT Robert E. Scurlock Jr. 1001 Bradford Lane Fairfield Glade, Tennessee 38555	2